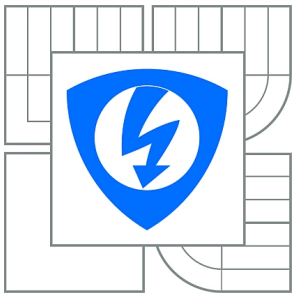
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

METODY PRO ZPRACOVÁNÍ SEGMENTOVANÝCH OBRAZŮ

METHODS FOR SEGMENTED IMAGE PROCESSING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

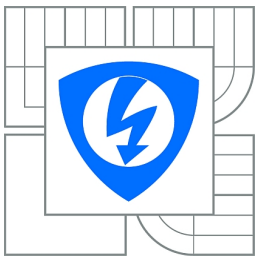
AUTOR PRÁCE — Bc. RADEK ŠTĚRBA

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN ZUKAL

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Radek Štěrba	***ID:***	72859
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Metody pro zpracování segmentovaných obrazů

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a vytvořte vhodnou datovou strukturu pro popis jednotlivých segmentů a jejich vzájemných vztahů v obraze. Dbejte na to, aby bylo možné uchovávané informace o segmentech v budoucnu libovolně měnit či přidávat. Dále navrhněte a zrealizujte algoritmus, který tuto strukturu ze segmentovaného obrazu vytvoří. Navrhněte a vytvořte také potřebné nástroje pro grafické zobrazení vámi navrženého řešení.

DOPORUČENÁ LITERATURA:

[1] Bovik, Al (ed.). Handbook of Image and Video Processing. San Diego: Academic Press, 2000. ISBN 0121197905

[2] WU, Z; LEAHY, R. An optimal graph theoretic approach to data clustering : theory and its application to image segmentation. In IEEE Transactions on : Pattern Analysis and Machine Intelligence. [s.l.] : [s.n.], 1993. s. 1101 - 1113 . Dostupné z WWW: <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=244673>. ISSN 0162-8828

[3] YI-HUA, Lan, et al A novel image segmentation method based on random walkcuts and image segmentation. In PACIIA 2009. Asia-Pacific Conference on : Computational Intelligence and Industrial Applications. [s.l.] : [s.n.], 2009. s. 207-210. Dostupné z WWW: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5406455&isnumber=5406359>. ISBN 978-1-4244-4606-3.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Martin Zukal

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

Anotace

Tato práce se zabývá reprezentací segmentovaných obrazů pomocí grafů. Také jsou zde popsány různé segmentační techniky, které se používají při zpracování obrazové informace. Matematika je v dnešní době stále více zapotřebí. V této práci tato skutečnost není opomenuta. Jsou zde popsány základní informace o teorii.

Druhá část práce je praktická. Obsahuje průzkum knihoven zpracovávajících grafy. Dále jsou zde popsány datové struktury pro popis segmentovaného obrazu. V neposlední řadě je zde také popsána tvorba a vlastnosti operátorů pro prostředí RapidMiner, které tyto struktury plní.

Klíčová slova

Segmentace, oblast, hrana, graf, datový typ, knihovna, RapidMiner, operátor, obraz.

Abstract

This work deals with the representation of segmented images using graphs. Different segmentation methods used in processing visual information are described here. Today is mathematics increasingly needed. This fact is not omitted, basic information of graph theory are described in this paper.

The second part of this work is practical. It contains a survey of libraries processing graphs. Further the data structures for describing the segmented image are described. Last but not least, there is also described the formation and properties of the operators designed for environment RapidMiner that fill these structures.

Key Words

Segmentation, area, edge, graph, data type, library, RapidMiner, operator, image.

ŠTĚRBA, R. *Metody pro zpracování segmentovaných obrazů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 64 s. Vedoucí diplomové práce Ing. Martin Zukal.

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma Metody pro zpracování segmentovaných obrazů jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

Poděkování

Rád bych poděkoval vedoucímu mé diplomové práce Ing. Martinu Zukalovi za odbornou pomoc a vedení při zpracování mé diplomové práce.

V Brně dne .....

.....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době je zpracování obrazové informace velmi potřebné. Mnoho odvětví by se bez něj neobešlo, např. kriminalistika (např. rozpoznávání tváře hledané osoby v davu), průmyslová výroba či zábavní průmysl. To je důvod, proč se tato práce věnuje zpracování obrazu.

Ovšem zpracování obrazu je velice široký pojem. Tato práce se věnuje jen části této disciplíny, je to pouze střípek v mozaice. Následující text se zabývá např. otázkami: Lze nějakým způsobem reprezentovat segmentovaný obraz? A pokud ano, jakým způsobem?

Naskýtá se ovšem otázka: K čemu je popis jednotlivých objektů v obraze? Každý předmět z reálného světa lze popsat tak, aby byl nezaměnitelný s jinými předměty. Pokud z obrazu získáme dostatečný počet informací, tak můžeme zjistit, jaké předměty se nachází v obraze. Tyto algoritmy mohou být uplatněny následně i při zpracování video sekvencí, což je vlastně sled obrázků. Budeme pak moci detekovat i pohyb jednotlivých objektů. Toto může značně urychlit nebo také zautomatizovat práci, kterou dnes zastává člověk (např. dozorce u kamerového systému už nebude zapotřebí, nebo alespoň vyloučí lidskou chybu – v tomto případě usnutí či nepozornost).

Důležitým faktorem při zpracování je, jakým způsobem lze implementovat tuto myšlenku. Jestli to bude samostatný program, či zásuvný modul pro nějaké prostředí? Pokud již existuje nějaké prostředí, které se zmíněnou problematikou zabývá, tak by bylo zbytečné a více pracné vytvářet takovýto samostatný program.

Techniky pro zpracování obrazu se neustále vyvíjejí, takže i tento program, se bude vyvíjet (budou se přidávat další vlastnosti, kód programu se bude optimalizovat, …), takže programátor by měl při vývoji myslet především na ty, kdo budou jeho dílo dále využívat. Takže je velmi důležité, jakým způsobem se budou uchovávat získaná data.

Výše zmíněné otázky i další, doposud nevyřčené, se tato práce pokusí zodpovědět.

1 Segmentace

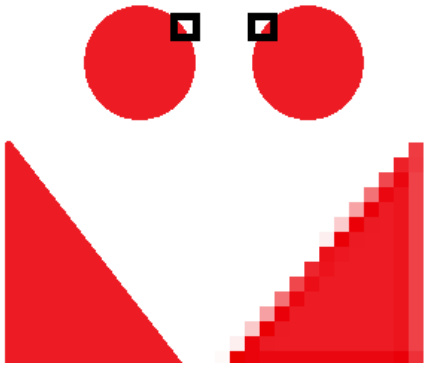
1.1 Digitální obraz

Digitální obraz je dvourozměrnou reprezentací zachycené scény. K uložení těchto informací se používá binární soustava. [1]

Rozlišují se dva druhy digitálních obrazů:

- vektorový,
- rastrový.

Vektorový obraz je popsán základními geometrickými útvary (bod, přímka, křivka a mnohoúhelníky), které jsou založeny na matematických rovnicích. Takto reprezentovaný obraz může dosáhnout téměř jakéhokoli rozlišení a útvary v obraze lze spravovat jednotlivě. Vektorová grafika se používá spíše na jednoduché ilustrace (např. logo). U složitějších obrazů je zpracování hardwarově vysoce náročné. [2]



Obr. 1.1 Vektorový a rastrový obraz

Rastrový obraz je popsán pomocí barevných bodů – pixelů. Každý pixel je definován polohou v obraze a barevnými složkami např. složkami RGB modelu (Red Green Blue). Zmíněné obrazové body jsou uspořádány do matice. Tyto obrazy se využívají pro fotografie či video. Oproti vektorové grafice má rastrová menší nároky na výpočetní techniku, ale kvalita obrazu je při zvětšení horší. [2]

Na obr. 1.1 je vidět rozdíl mezi vektorovým (vlevo) a rastrovým (vpravo) obrazem.

1.2 Co je segmentace

Segmentace je důležitý krok procesu analýzy obrazu. Cílem segmentace je rozdělit obraz na jednotlivé části (segmenty), které jsou velmi podobné objektům v reálném světě, jež jsou obsaženy v obraze. [3]

Segmentace může být úplná, tzn. rozdělení obrazu na nepřekrývající se části shodující se s objekty skutečného světa. Nebo může být pouze částečná, kde segmenty obrazu přímo nekorespondují s objekty v obraze (tyto segmenty mohou být použity v pozdější analýze). K dosažení úplné segmentace je nutná součinnost s vyšší úrovní zpracování, jež využívá specifických znalostí problematiky. [3]

V prvotní fázi zpracování obrazu není možné dosáhnout naprosto správné a úplné segmentace komplexních scén. Rozumné by bylo použít částečnou segmentaci jako vstup pro vyšší úroveň zpracování. [3]

Nejednoznačnost obrazových dat je jedním ze zásadních problémů segmentace. Z toho důvodu se segmentace dělí do tří skupin podle toho, jaký nástroj se pro segmentaci využívá:

- prahování (využívá histogram),
- detekce hran,
- detekce oblastí.

U posledních dvou skupin lze použít více vlastností resp. charakteristik jako je jas nebo struktura segmentu. Také tyto skupiny řeší stejný problém, protože každý region může být určen svou hranicí a také každá hranice popisuje region. Ovšem při použití těchto metod se nemusí vždy dostat stejný výsledek. [3]

1.3 Metody segmentace

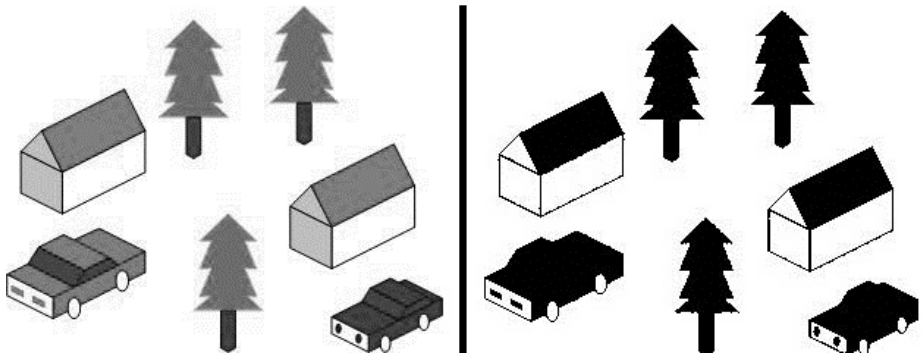
1.3.1 Prahování

Prahování podle stupňů šedi je jedna z nejjednodušších a nejstarších segmentačních technik. Pro mnoho objektů v obraze je charakteristické konstantní odrazivostí nebo

absorpcí světla. Prahování je výpočetně nenáročné, segmentace probíhá téměř v reálném čase, a to je jedním z důvodů, proč se tato metoda stále používá. Tato metoda se hodí pro případy, kdy se jednotlivé objety vzájemně nedotýkají a jsou zřetelně odlišné od pozadí. [3]

Prahování může probíhat následovně: Prochází se celý obraz pixel po pixelu a zkoumá se, zda hodnota (jasová) pixelu je či není větší než hodnota prahu. Pokud je větší, tak se danému pixelu přiřadí hodnota 255, a pokud je menší, tak se mu přiřadí hodnota 0. Výsledný obraz je tudíž černobílý. Ukázka prahování je vidět na obrázku obr. 1.2, byla použita hodnota prahu 150. [3]

Výběr prahu je kritický. Pokud bude vybrána špatná hodnota, nemusí být výsledek segmentace uspokojivý. Použít jedinou hodnotu prahu pro celý obraz je vhodné pouze za speciálních podmínek (např. pro jednoduché obrázky). Obrazy obvykle nemají rovnoměrné rozložení barev (jak objekty, tak pozadí). Za toto rozložení může například nerovnoměrné osvětlení scény. Proto se požívá prahování s proměnnou hodnotou prahu (také se nazývá adaptivní prahování). Hodnota prahu se získává pomocí funkce, jež je vytvořena na základě lokálních obrazových charakteristik. Tzn. použije se několik prahů, které jsou pozičně závislé. [3]



Obr. 1.2 Ukázka prahování

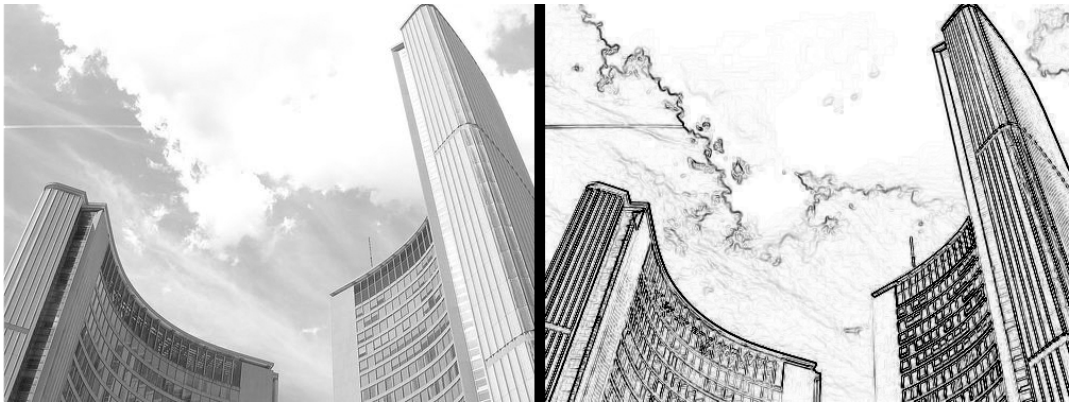
Další variací je tzv. poloviční prahování (semitresholding). Účelem tohoto procesu je odfiltrovat pozadí a nechat informace objektů nedotčené. [3]

Existuje více metod pro určování prahu, ty lze nalézt v [3].

1.3.2 Detekce hran

Název segmentace pomocí detekce hran zastupuje velkou skupinu metod zakládajících se na informacích o hranách v obraze. Tento typ segmentace závisí na operátorech detekujících hrany v obraze. [3]

Hrany značí v obraze jistou nespojitost. Tato nespojitost může být ve stupni šedi, v barevné škále, v texturách atd. Tyto nespojitosti jsou hledány již výše zmíněnými operátory. Dalším krokem musí být pospojování detekovaných hran do řetězce pro vznik hranic v obraze. V poslední řadě je nutné provést *vlastní segmentaci*, neboli musí se vybrat pouze ty zřetězené hrany, které mají souvislost se skutečným světem, a následně prezentovat v obraze, který bude obsahovat pouze tyto hrany. [3]



Obr. 1.3 Ukázka segmentace pomocí detekce hran

Existuje také několik metod segmentace, u kterých dostaneme jiné hranice tudíž i jiný výsledek. Tyto metody se liší strategií a také množstvím dřívějších informací, které v metodě mohou být použity. Čím více těchto informací dodáme, tím přesnější bude segmentace. [3]

Závažným problém pro segmentaci s detekcí hran je šum v obraze a nežádoucí informace v obraze. Při výskytu těchto nežádoucích jevů mohou vzniknout hranice v místech, kde ve skutečnosti žádné hranice nejsou. [3]

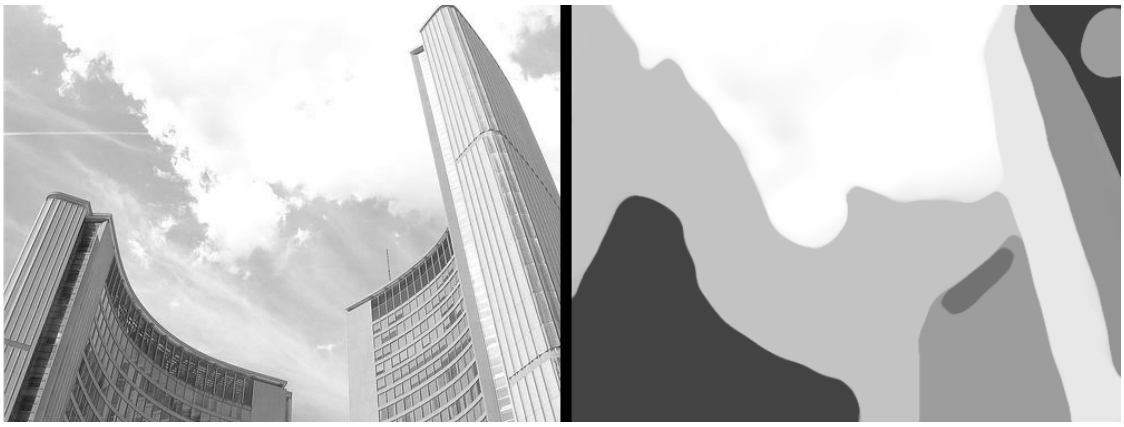
Metod využívající detekci hran je velké množství např. hranové prahování, prahování hran a sledování hranice. Ukázka segmentace je zobrazena na obr. 1.3. Více informací k těmto i dalším metodám lze nalézt v [3].

1.3.3 Detekce oblastí

Metody, které byly popsány v kap. 1.3.2, nacházely hranice mezi oblastmi. Následující metoda vytváří oblasti přímo. Výsledky při použití segmentace využívající detekci hran a segmentace pomocí detekce oblastí nejsou obvykle stejné. Pro lepší výsledek se mohou výstupy těchto metod kombinovat. Segmentační techniky využívající detekci oblastí jsou obecně lepší v obrazech obsahujících šum (v těchto obrazech je velice obtížné detekovat hranici). [3]

Důležitá vlastnost jednotlivých oblastí je homogenita. Ta se používá jako hlavní kritérium pro segmentaci využívající detekci oblastí. Z toho vyplývá, že hlavní myšlenkou segmentačních technik, které jsou popisovány v této kapitole, je rozdělit obraz na oblasti s největší homogenitou. Kritéria pro homogenitu mohou být různá (např. barevný odstín, intenzita jasu nebo tvar – podoba na předmět). [3]

Obraz je správně rozdělen na segmenty, pokud jsou všechny oblasti homogenní a zároveň neexistují-li žádné dva segmenty, které by byly při spojení homogenní. [3]



Obr. 1.4 Segmentace s použitím detekce oblastí

Ukázku segmentace s využitím popisované metody lze vidět na obr. 1.4. Výstup této metody segmentování je vhodný pro další zpracování, z pohledu popisu obrazu i jednotlivých objektů v něm obsažených.

1.4 Shrnutí

Rozlišují se dva druhy digitálních obrazů: vektorový (popsán základními geometrickými útvary, obraz dosahuje větší kvality na úkor hardwarové náročnosti) a rastrový (popsán jednotlivými body, které nesou informaci o pozici a barvě).

Segmentace rozděluje obraz na části podle určitých kritérií. Metody pro segmentaci se dělí do tří skupin: prahování, detekce hran a detekce oblastí. Výsledky segmentace těchto metod se mohou lišit, a proto by se každá metoda měla používat na jiný obraz. Výstupy posledních dvou zmíněných metod se mohou kombinovat k dosažení lepšího výsledku.

2 Teorie grafů

Graf je abstraktní reprezentace souboru objektů, kde některé objekty jsou vázány jistou závislostí k jiným objektům. Tento typ datové abstrakce se hojně využívá v rozličných odvětvích zvláště v informatice.

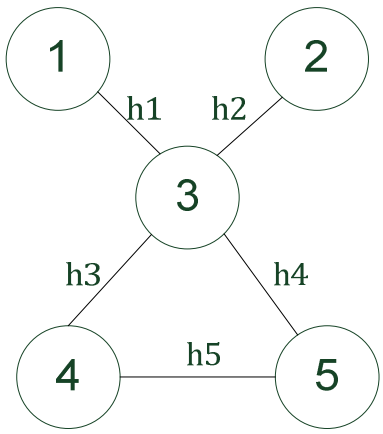
2.1 Definice

Obecný graf (dále jen graf) je uspořádaná dvojice $G = (V, H)$, kde V reprezentuje množinu vrcholů (též uzlů) a H množinu hran. Graf, který je rozšířen o hodnocení hran, se nazývá ohodnocený graf. Tento vážený graf je upořádaná trojice $G = (V, H, W)$, kde W představuje množinu vah. [4]

Graf, který má hrany nebo vrcholy popř. obojí opatřeny číselnými popř. jinými hodnotami, se nazývá ohodnocený graf. [5]

Hrana je spojnice právě dvou vrcholů. Vrcholy spojené touto hranou jsou sousední vrcholy. Grafy se mohou zadávat graficky (obr. 2.1) nebo výčtem jednotlivých prvků [4]:

$$\mathrm{V} = \{1,2,3,4,5\},\ \mathrm{H} = \big\{\{1,3\},\{2,3\},\{3,5\},\{3,4\},\{4,5\}\big\}.$$



Obr. 2.1 Ukázka grafu

2.2 Užitečné pojmy

Stupeň vrcholu

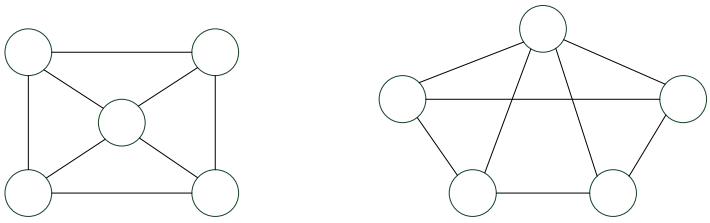
Pojmem stupeň vrcholu se rozumí počet hran vycházejících z příslušného vrcholu. Stupně vrcholů pro graf na obr. 2.1 jsou vzestupně 1, 1, 2, 2, 3. Nejvyšší resp. nejnižší stupeň v grafu G je stupeň s nevyšší resp. nejnižší hodnotou stupně. Nejvyšší stupeň vrcholu se značí $\Delta(G)$ a nejnižší stupeň vrcholu $\delta(G)$. [4]

Cesta

Cesta je sled jednotlivě spojených vrcholů, kde se žádný vrchol neopakuje. Pokud se berou v potaz jen vrcholy cesty, tak žádný vrchol nesmí mít větší stupeň než dva. Zvláštní případ cesty je tzv. Hamiltonova cesta, tato cesta grafu G obsahuje všechny uzly grafu G. [4]

Vzdálenost vrcholů

Vzdálenost dvou vrcholů v grafu je dána délkou nejkratší posloupnosti mezi těmito vrcholy v grafu G. Pro graf, který není vážený, se tato vzdálenost rovná počtu přeskoků na cestě mezi těmito uzly. Pro vážený graf se vzdálenost rovná součtu ohodnocení jednotlivých hran na cestě mezi dvěma příslušnými vrcholy. [4]



Obr. 2.2 Izomorfizmus

Izomorfizmus

Dva grafy jsou izomorfní, pokud se liší jiným číslováním uzlů a jiným zobrazením nebo obojím. Váhy jednotlivých hran musí ovšem zůstat nezměněny. Ukázka dvou izomorfních grafů je na obr. 2.2. [5]

Podgraf

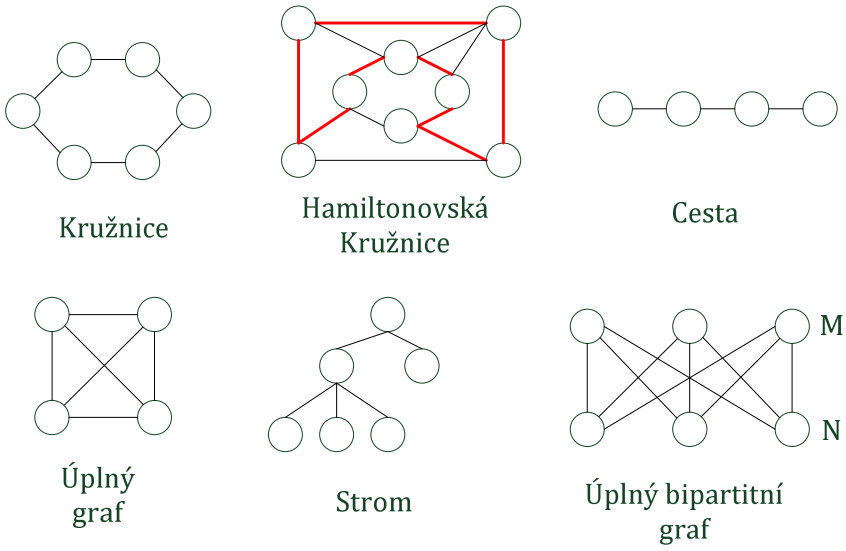
Určitý graf $G' = (V', H')$ je podgrafem grafu $G = (V, H)$, pokud V' resp. H' je podmnožinou V resp. H. Tedy graf G' musí být podmnožinou grafu G. [6]

2.2.1 Speciální typy grafů

Úzus je takový, že některé grafy se nazývají jmény, která je popisují. Jsou to tyto grafy:

- Kružnice – musí mít čtyři a více vrcholů, jež jsou spojeny hranami do cyklu. Jinými slovy kružnice je každá uzavřená cesta. [4]
- Hamiltonovská kružnice – v grafu G je podgraf, který je izomorfní a obsahuje všechny vrcholy grafu G. Neboli Hamiltonovská kružnice je speciální druh kružnice, která prochází každým vrcholem grafu právě jednou. Graf, který obsahuje tuto kružnici, se nazývá Hamiltonovský graf. [4]
- Cesta – již bylo popsáno výše.
- Úplný graf – v tomto grafu je každý vrchol spojen se všemi ostatními vrcholy příslušnou hranou. [4]
- Úplný bipartitní graf – má vrcholy rozděleny do dvou skupin (M a N). Počet prvků v každé skupině musí být větší nebo roven jedné. Každý prvek ze skupiny M resp. N je spojen s každým prvkem ze skupiny N resp. M a zároveň žádný prvek jedné skupiny není spojen s prvkem téže skupiny. [4]
- Strom – jednoduchý graf, který neobsahuje kružnice. U tohoto grafu existuje pouze jedna cesta mezi jednotlivými vrcholy. Tento typ grafu je často používán v informatice pro uchovávání informací. [7]

Ukázka popsaných grafů je na obr. 2.3.



Obr. 2.3 Speciální typy grafů

2.3 Reprezentace grafu

Aby bylo možné obecné grafy zpracovávat pomocí počítače, lze je reprezentovat třemi způsoby. Jsou to tyto:

- incidenční matice,
- matice sousednosti,
- spojový seznam.

Dále budou popsány první dva zmíněné. Informace o spojových seznamech lze nalézt v [5].

2.3.1 Incidenční matice

Incidenční matice (Incidence matrix) $M(G)$ je matice, kde řádky značí vrcholy grafu a sloupce hrany grafu. Tedy rozměr matice je $|V|$x$|H|$. Zapisuje se do ní následovně: pokud existuje hrana mezi zkoumaným vrcholem a jiným vrcholem, tak se do matice zapíše jednička (vrchol udává řádek a hrana sloupec). Incidenční matici pro graf zobrazen na obr. 2.1 je vidět na obr. 2.4. [5]

	h1	h2	h3	h4	h5
1	1	0	0	0	0
2	0	1	0	0	0
3	1	1	1	1	0
4	0	0	1	0	1
5	0	0	0	1	1

Obr. 2.4 Incidenční matice

Do incidenční matice pro orientovaný graf se zapisuje poněkud odlišně. Pokud příslušná hrana z vrcholu vystupuje tak se do příslušné pozice zapíše hodnota „1“, pokud ovšem hrana do vrcholu vstupuje, tak se zapíše hodnota „ – 1,“ jinak se zapisuje hodnota „0.“ Neboli do matice $M(G) = (m_{VH})$ se zapisuje dle předpisu [8]:

$$m_{VH} = \begin{cases} +1 & \text{pokud hrana } H \text{ vychází z vrcholu } V \\ -1 & \text{pokud hrana } H \text{ vchází do vrcholu } V \\ 0 & \text{jinak} \end{cases} \cdot \tag{2.1}$$

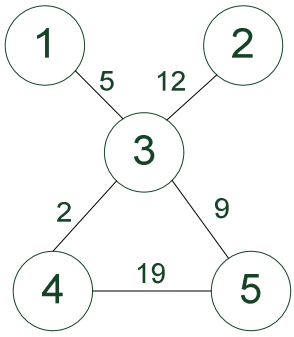
2.3.2 Matice sousednosti

V matici sousednosti (Adjacency matrix) $A(G)$ značí řádky i sloupce vrcholy grafu. Tato matice má rozměr $|V|$x$|V|$. Pokud se v grafu vyskytne hodnota jedna, tak souřadnice udávají, které dva vrcholy jsou spojeny. Matice sousednosti je na rozdíl od incidenční matice souměrná podle diagonály (ukázka na obr. 2.5). [5]

Matice nemusí být vyplněny pouze binárními hodnotami, ale na místě „1“ může být ohodnocení hrany, pokud jsou hrany ohodnoceny. Do matice sousednosti $A(G) = (a_{VV'})$ se může tedy zapisovat dle následujícího přepisu:

$$a_{VV'} = \begin{cases} W_{VV'} & \text{pokud mezi vrcholy } V \text{ a } V' \text{existuje hrana } H_{VV'} \\ 0 & \text{jinak} \end{cases}, \tag{2.2}$$

kde V a V' jsou vrcholy grafu a $W_{VV'}$ je ohodnocení hrany $H_{VV'}$ mezi příslušnými vrcholy. Ukázka matice sousednosti je na obr. 2.5. [5]



	1	2	3	4	5
1	0	0	5	0	0
2	0	0	12	0	0
3	5	12	0	2	9
4	0	0	2	0	19
5	0	0	9	19	0

Obr. 2.5 Matice sousednosti

2.4 Prohledávání grafu

Pokud se člověk dívá na graf, vnímá ho jako obraz a tedy jeho vnímání grafu je „globální.“ U zpracovávání grafu počítačem je toto vnímání jiné. Počítač jej nevnímá globálně (jako celek), ale lokálně (prvek po prvku). To působí problémy při prohledávání grafu. Metody prohledávání grafu jsou tyto [4]:

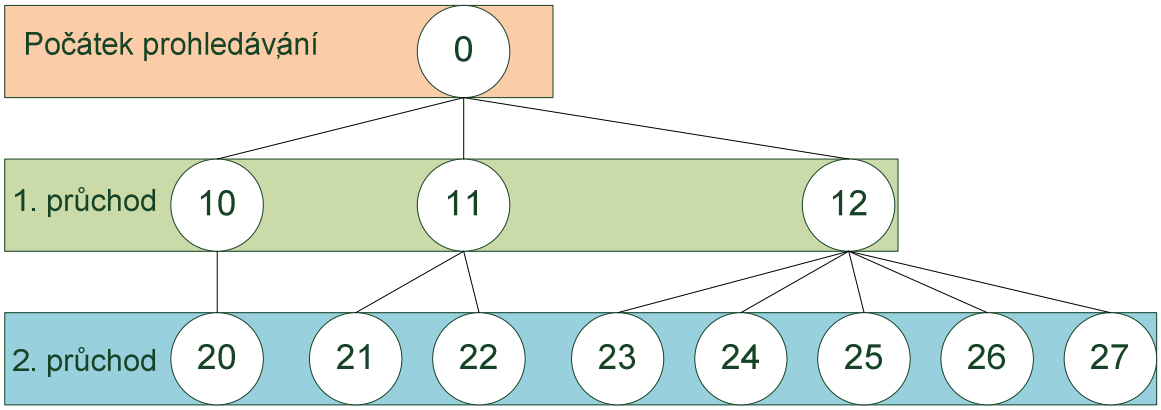
- neinformované metody – tyto metody postupně prochází graf a neřídí se žádnou prioritou pro procházení jednotlivých vrcholů, zástupci těchto metod jsou prohledávání do šířky (breadth-first search – BFS) a prohledávání do hloubky (depth-first search – DFS);

- informované metody – tyto metody na rozdíl od předchozích preferují při procházení některé vrcholy před jinými, zástupcem těchto metod je Dijkstrův algoritmus.

2.4.1 Prohledávání do šířky

Prohledávání do šířky, je takový algoritmus, který postupně prochází všechny vrcholy grafu. BFS nejdříve prochází vrcholy blízké počátku prohledávání, tzn. nejdříve projde vrcholy sousedící s počátkem prohledávání (vzdálenost počátku a prohledávaných uzlů je jedna). Počátek prohledávání je vrchol grafu, od kterého hledání začíná. [4]

Pokud BFS nenalezl řešení, tak následuje další průchod algoritmu grafem. V tomto průchodu projde všechny uzly, které mají vzdálenost od počátku hledání dva. Hodnota vzdálenosti se inkrementuje do té doby, kdy je nalezeno řešení nebo byly projity všechny vrcholy grafu. Ukázka algoritmu BFS je zobrazena na obr. 2.6. [4]



Obr. 2.6 Algoritmus BFS

Algoritmus BFS je úplný, tj. pokud existuje řešení, tak jej nalezne. Ovšem tento algoritmus má vysoké nároky na výpočetní paměť. BFS má vysokou časovou složitost $O(|V| + |H|)$, kde $|V|$ je počet vrcholů a $|H|$ je počet hran. [5]

2.4.2 Prohledávání do hloubky

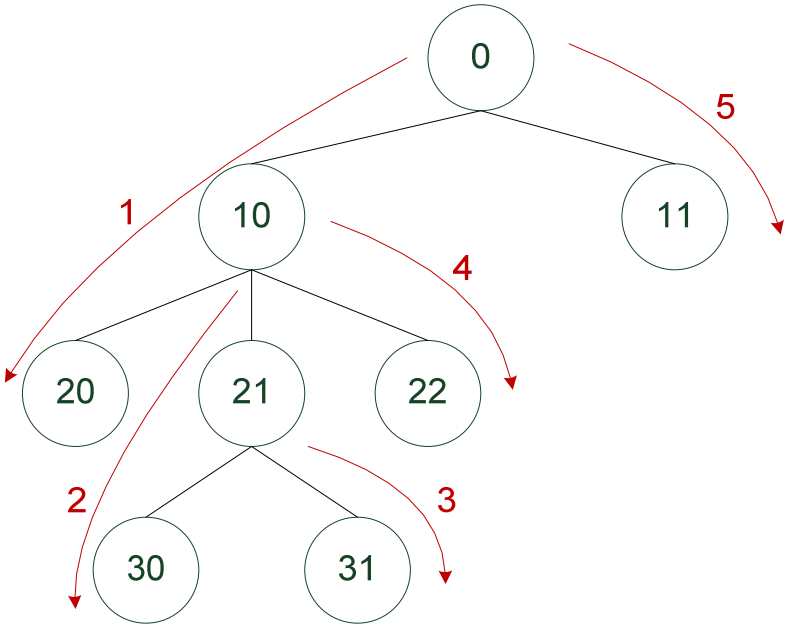
Algoritmus DFS prochází vrcholy grafu a upřednostňuje ty vrcholy, které jsou nejdále od počátku prohledávání. DFS pracuje následovně:

1. DFS projde nejdříve vrchol S, kde začíná procházení.
2. DFS projde prvního potomka P_1 uzlu S (pokud již tento potomek nebyl projit).
3. Uzel P_1 se označí jako S a algoritmus pokračuje od kroku 2.

Pokud algoritmus narazí na vrchol, který nemá potomka, tak se algoritmus vrací k nejbližšímu uzlu, který byl projit a má neprojitého potomka. Algoritmus skončí, pokud nalezne řešení nebo již byly projity všechny uzly. [4] [5]

Ukázka průchodu grafem je na obr. 2.7. Pořadí uzlů z obr. 2.7ve kterém byly projity je následující: {0}, {10}, {20}, {21}, {30}, {31}, {22}, {11}.

Algoritmus DFS je úplný. Časová složitost DFS je stejná jako u algoritmu BFS $O(|V| + |H|)$. [4]



Obr. 2.7 Algoritmus DFS

2.4.3 Dijkstrův algoritmus

Dijkstrův algoritmus hledá nejkratší cestu mezi dvěma vrcholy. Pro správnou funkci algoritmu musí mít prohledávaný graf ohodnoceny hrany nezápornými čísly. [5]

Algoritmus postupně zpřesňuje odhad nejkratší vzdálenosti od zdroje k ostatním vrcholům. $l(C)$ je délka nejkratší cesty z vrcholu Z (zdroj) do vrcholu C (cíl). Dále se uvažuje, že $H(u,c)$ je hrana ohodnocená $W(U,C)$. Aktuální odhad nejkratších vzdáleností uzlům U a C je $l(U)$ a $l(C)$. Takže pokud platí

$$l(U) + W(U,C) < l(C), \tag{2.3}$$

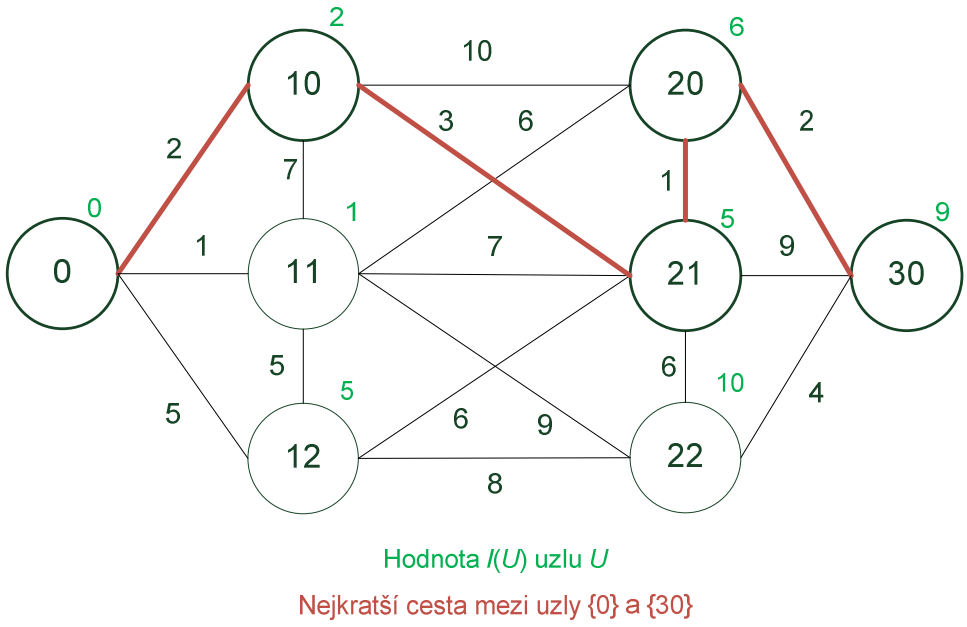
potom

$$l(C) = l(U) + W(U,C), \tag{2.4}$$

tj. hodnota výrazu $l(U) + W(U,C)$ se stává novým odhadem délky l(C). Tento proces se nazývá relaxace. [5]

Pokud se relaxace provede opakovaně nad všemi hranami grafu, pak odhady $l(C)$ se blíží k délkám nejkratších cest od Z do C. Množina $\{Z\}$ je množina, pro niž je již zjištěna konečná hodnota $l(C)$. Tato množina $\{Z\}$ je před spouštěním algoritmu prázdná. Množina se plní takto: Dijkstrův algoritmus v každém kroku vybere z dvojice $[Z, V]$ takový vrchol U, který má nejmenší $l(U)$. Tato strategie se nazývá chamtivá (greedy). [5]

Časová složitost algoritmu je $O(|V|^2)$. Ukázka Dijkstrova algoritmu je vidět na obr. 2.8. [5]



Obr. 2.8 Dijkstrův algoritmus

Další algoritmy pro prohledávání grafu lze najít v [4] a [5].

2.5 Shrnutí

Obecný graf je uspořádaná dvojice $G = (V, H)$, kde V zastupuje množinu vrcholů (též uzlů) a H množinu hran. Graf může mít hrany i vrcholy opatřeny libovolnými hodnotami.

Stupeň vrcholu je počet hran uzlu. Vzdálenost vrcholů je počet přeskoků mezi uzly nebo součet hodnocení hran na cestě mezi vrcholy. Důležitá vlastnost grafů je izomorfizmus.

Graf pro počítačové zpracování může být reprezentován spojovým seznamem a maticemi (incidenční a maticí sousednosti). Matice sousednosti se jeví být dobrým řešením pro reprezentaci grafu v počítači.

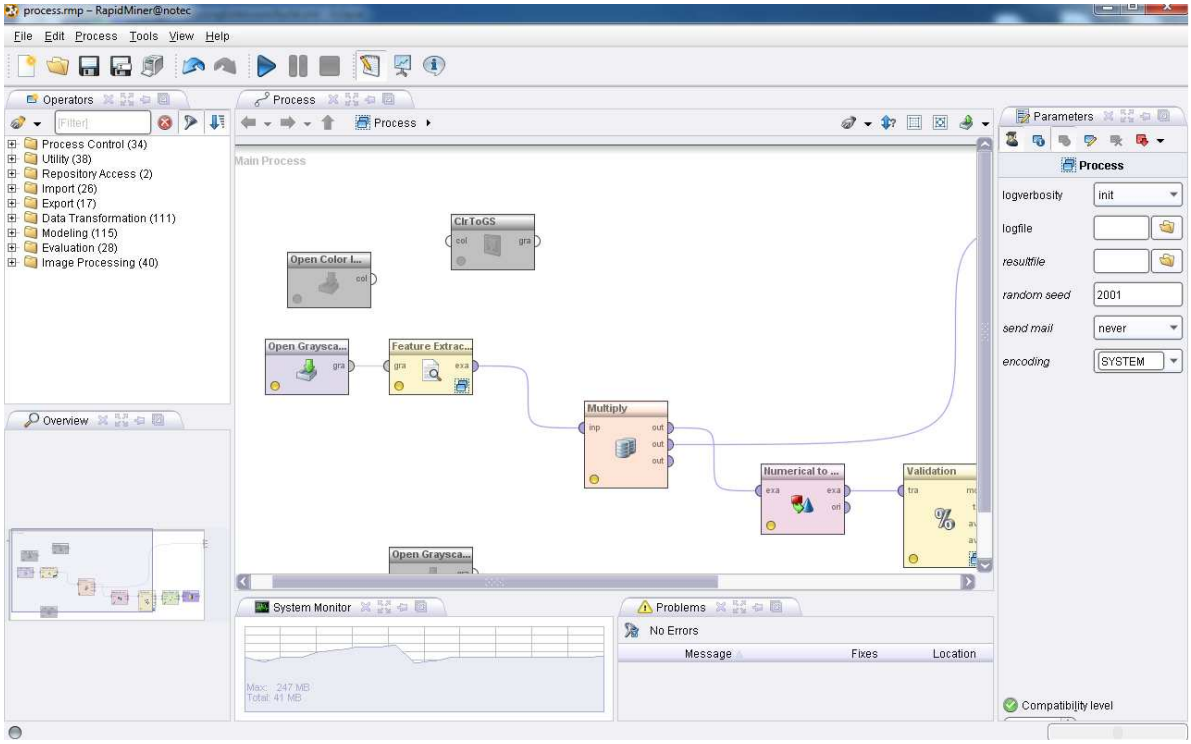
Počítač vnímá graf spíše lokálně, proto byly definovány algoritmy pro jeho procházení. Algoritmy BFS a DFS jsou relativně lehce implementovatelné, avšak jejich časová náročnost může být veliká. Oproti tomu Dijkstrův algoritmus je už složitější, ale ve většině případů má menší časovou složitost.

3 RapidMiner

RapidMiner (obr. 3.1) je prostředí pro strojové učení a tzv. data mining procesy. Koncepce modulárních operátorů umožňuje vytvořit komplexní řetěz operátorů pro řešení velkého počtu problémů.[9]

RapidMiner je graficky orientované prostředí. Jednotlivé bloky se skládají za sebe. Do některých bloků lze vložit další bloky. Takto pospojované operátory se nazývají strom operátorů. V tomto stromě listy představují jednoduché kroky v modelovém procesu, vnitřní uzly představují komplexnější nebo abstraktní kroky. [9]

Každý operátor definuje očekávaný vstup a svůj výstup. Některé operátory mají vstup pro upřesňující uživatelské parametry (obr. 3.1 vpravo).[9]



Obr. 3.1 RapidMiner – Grafické prostředí

RapidMiner využívá XML (eXtensible Markup Language) pro popis stromu operátorů. XML je značkovací jazyk, který se stal standardem pro popis výměny dat. XML je také snadno čitelný pro uživatele i pro stroje. Každý proces v RapidMineru je popsán pomocí XML.[9]

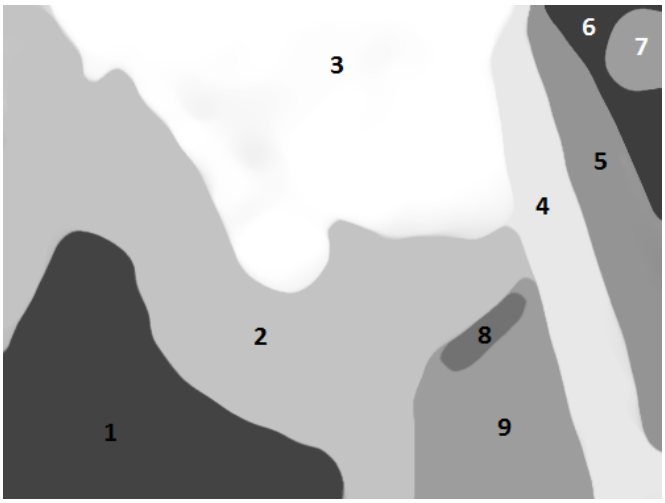
RapidMiner je distribuován pod svobodnou licencí AGPL (Affero General Public License).[10]

3.1 Shrnutí

RapidMiner je prostředí pro strojové učení a tzv. data mining procesy. RapidMiner je graficky orientované prostředí a jednotlivé operátory se skládají za sebe (pospojované operátory – strom operátorů).

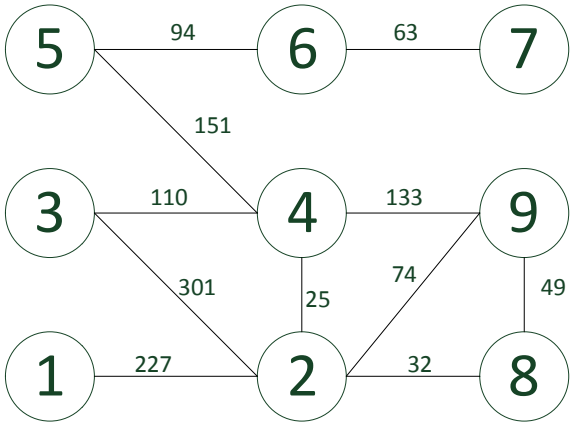
4 Specifikace výstupu

Na obr. 4.1 je vidět segmentovaný obraz. Jednotlivým segmentům jsou přiřazeny tzv. ID (identifikátory), které budou jednotlivé segmenty zastupovat.



Obr. 4.1 Segmentovaný obraz s popisem segmentů pomocí ID

Dále musí být vytvořen ohodnocený graf, který bude popisovat vztahy mezi jednotlivými segmenty. Graf, který popisuje obr. 4.1, je na obr. 4.2. Uzly zastupují jednotlivé segmenty. Hrany mezi dvěma uzly značí, že segmenty, které jsou zastoupeny těmito uzly, mají společnou hranici. Ohodnocení této hrany znamená délku společné hranice. Uzly také mohou obsahovat další informace o segmentech, které budou popsány v jiné kapitole.



Obr. 4.2 Ohodnocený graf popisující segmentovaný obraz[1]

[1] Ohodnocení jednotlivých hran nepopisuje přesně obr. 4.1. Hodnoty jsou pouze orientační.

4.1 Shrnutí

Výstupem by měl být ohodnocený graf, který bude popisovat vztahy mezi jednotlivými segmenty obrazu.

5 Tvorba grafu

Jak již bylo dříve řečeno, z extrahovaných dat bude vytvořen graf. Otázkou však zůstává, jakým způsobem tento graf vytvořit. Jistě by šlo vytvořit vlastní strukturu pro graf, ale tento způsob se zdá být poněkud zbytečný, protože už bylo vytvořeno množství knihoven, jak logických tak grafických, pro práci s grafy. Lepší způsob tedy bude vybrat vhodnou knihovnu resp. knihovny a s ní resp. s nimi pak pracovat.

5.1 Knihovny

V této kapitole bude představeno několik knihoven, které by mohli být použity při vytváření a zpracovávání grafu. Také zde budou popsány jejich vlastnosti a možnosti využití.

5.1.1 yFILES

Projekt yFILES, který vyvíjí německá společnost yWorks, umožňuje editovat (o tuto záležitost se stará editor Graphity) i prohlížet (prohlížeč yEd) velké množství struktur – biochemické sítě, UML diagramy, databáze, sociální sítě atd. Mj. je tento projekt přizpůsoben pro práci s grafy a má implementovány základní operace s nimi.[11]

Část projektu, která se hodí k této práci je yFILES for Java, což je knihovna určená pro Javu, která obsahuje funkce pro vytváření a, což je podstatné, pro vizualizaci výše zmíněných struktur.[11]

Ovšem knihovna yFILES for Java není šířena pod open source licencí. Tato knihovna je společností yWorks vydávána pod licencí shareware. Může být bezplatně používána po dobu 30 dnů.[11]

5.1.2 Java Universal Network/Graph Framework (JUNG)

JUNG je softwarová knihovna, která poskytuje rozšiřitelný jazyk pro modelování, analýzu a vizualizaci dat, která mohou být reprezentována grafem nebo sítí. Tato knihovna je napsána v jazyce Java. [12]

JUNG podporuje značné množství grafů jako např. orientované a neorientované grafy, grafy s paralelními hranami, hypergrafy (hypergraphs). Knihovna má také implementováno jisté množství algoritmů pro práci s grafovými strukturami, data mining a analýzu sociálních sítí – nalezení nejkratší cesty pomocí Dijkstrova algoritmu, nástroje pro optimalizaci, generování náhodného grafu a další.[12]

Knihovna JUNG je poskytována pod licencí BSD (Berkeley Software Distribution).Tato licence umožňuje volné šíření licencovaného obsahu a vyžaduje pouze uvedení autora a licence s upozorněním odpovědnosti za dílo. Kompletní licence projektu je v lit. [13].

5.1.3 Annas

Annas je open source framework, jež je napsán v jazyce Java. Tento nástroj je určen především pro pracovníky zabývající se výzkumem a vývojem na poli teorie grafů. Annas má v sobě implementováno množství grafových struktur, algoritmů pro práci s těmito strukturami, matematických funkcí aj. Projekt annas je složen ze dvou hlavních balíčků. Tyto balíčky jsou:

- Annas.Graph – balíček pro vytváření, manipulaci a vizualizaci grafových struktur,
- Annas.Math – balíček obsahující množství matematických funkcí.[14]

Díky frameworku annas lze používat rozličné datové struktury, jsou jimi např. orientovaný a neorientovaných graf, multigrafy a grafy jen pro čtení (read only). Dále také obsahuje metody pro exportování grafů v podobě xml souboru nebo v podobě pole obsahující matici sousednosti. Jak už také bylo zmíněno výše annas obsahuje funkce pro vykreslení grafové struktury.[14]

Annas je publikován pod licencí GNU GPL (General Public License – všeobecná veřejná licence), což je licence pro open source software. GNU GPL je známým příkladem copyleftové licence, tzn. odvozená díla musí být dostupná pod stejnou licencí jako dílo původní.[15]

5.1.4 G

G je rozšiřitelná generická grafická knihovna (generic graphics library), která je orientovaná na vizualizaci grafů a vytváření 2D grafických objektů. Tuto grafickou knihovnu vyvíjí GeoSoft. [16]

Knihovna G nabízí mimo jiné:

- vytváření hierarchických scén s grafy,
- rozšiřitelnost na 3D prostor,
- podporu rastrových obrázků,
- prostředky pro tvorbu a transformaci geometrických objektů,
- funkce pro detekci objektů.

Bohužel knihovna G je kompatibilní s JDK (Java Development Kit) do verze 1.5. Takže G neumí spolupracovat s nejnovější JDK.[16]

Grafická knihovna G je distribuována pod open source licencí GNU LGPL (Lesser General Public License). Tato licence je kompromis mezi GNU GPL a BSD. Copyleftové omezení se stahuje pouze na samotný software, nikoli na software, který jej používá.[17]

5.1.5 JGraph

JGraph je oblíbená knihovna pro vizualizaci grafů (grafů z teorie grafů, jak bylo popsáno v kap. 2). Vizualizace je pro návrháře této knihovny velmi důležitá, proto je rozsah funkcí pro vizualizaci nepřeberný. Vrcholy grafu mohou být různé tvary, obrázky, animace aj. [18]

Interakce je důležitá vlastnost, kterou knihovna JGraph vlastní. V zásadě jde o to, že vytvořený graf není pouze obrázek, ale aplikace, jež povoluje plnou interakci uživatele s vizualizací grafu. JGraph podporuje přemísťování, kopírování, změnu velikosti vrcholů, propojování a odpojování vrcholů, editování vrcholů atd. [18]

I když je JGraph knihovna zabývající se vizualizací, má v sobě zahrnuty jisté metody pro analýzu grafu např. zjištění nejkratší cesty mezi dvěma uzly. JGraph také podporuje řadu stromů a hierarchických struktur, které mohou být automaticky aplikovány na graf. [18]

JGraph je publikován pod licencí BSD. [18]

5.1.6 JGraphT

JGraphT je široce používaná open source knihovna, napsaná v jazyce Java, která se používá pro zpracování grafu. Tato knihovna tedy poskytuje matematické modely z teorie grafů (různé druhy grafů) a algoritmy pro pracování s těmito objekty. [19]

JGraphT, jak již bylo řečeno výše, podporuje množství rozličných grafových struktur. Jsou jimi například:

- orientovaný a neorientovaný graf,
- vážené a nevážené grafy (typ váhy může být definován uživatelem – např. číselná a jiná hodnota váhy),
- grafy s různými hranovými možnostmi, jsou jimi jednoduché grafy, multigrafy (dva vrcholy v tomto grafu mohou být spojeny více než jednou hranou) a pseudograf (neorientovaný graf, kde je dovoleno, aby uzly byly spojeny větším počtem hran, a aby některé uzly byly opatřeny smyčkami),
- grafy „read-only“ (pouze pro čtení), tyto grafy nelze modifikovat,
- podgrafy. [19]

Vrcholem grafu, vytvořeného pomocí knihovny JGraphT, může být jakýkoliv objekt. Graf se může vytvářet na základě řetězců `String`, URL (Uniform Resource Locator – jednotný lokátor zdrojů), XML dokumentů aj. Lze vytvářet i graf grafů (uzly jednoho grafu jsou celé grafy). [19]

Pomocí této knihovny lze již vytvořené grafy exportovat do souborů a datových struktur. Díky této možnosti exportu, lze tedy grafy, vytvořené pomocí této knihovny, zpracovávat nebo vizualizovat pomocí jiných nástrojů. Graf lze exportovat např. do:

- DOT souborů,
- souborů GML (Graph Modelling Language),
- souborů GraphML.[20]

Tato knihovna také umí vytvořit soubor `*.csv`, který lze importovat do programu MS Visio (MicroSoft Visio). Pomocí tohoto programu si tedy lze vytvořený graf vizuálně prohlédnout. Dále také tato knihovna umí z grafu vytvořit matici (např. matici sousednosti), tuto matici pak mohou zpracovávat programy určené pro zpracovávání matic. Představitelem těchto programů může být Matlab.[20]

Dále umí JGraphT spolupracovat s již výše zmíněnou knihovnou JGraph. Pomocí adaptéru `JGraphModelAdapter<V,E>` lze „logický“ graf vytvořený v knihovně JGraphT převést do grafické podoby knihovny JGraph.[20]

JGraphT má v sobě také zakomponováno množství algoritmů pro práci s grafovými strukturami. Ke komplexnějším patří:

- Dijkstrův algoritmus pro hledání nejkratší cesty (bylo popsáno v kap. 2.4.3),
- Floyd-Warshallův algoritmus, tento algoritmus hledá nejkratší cestu mezi dvěma uzly, tento algoritmus má časovou složitost $O(|V|^3)$,
- algoritmus řešící problém nalezení optimální nebo přibližně optimální nejkratší cesty mezi vrcholy (hamiltonův kruh), tento problém je obecně znám jako problém obchodního cestujícího
- a další.[20]

Knihovna obsahuje ovšem i méně složité algoritmy, jsou jimi např. funkce pro:

- zjištění sousedních vrcholů,
- detekci kruhů v grafu,
- výpočet tranzitivního uzávěru. [20]

Podrobné informace o zmíněných algoritmech lze nalézt v lit. [4] a [5].

Knihovna JGraphT stejně jako knihovna G je distribuována pod volnou neboli open source licencí GNU LGPL.[17]

5.2 Výběr

Je potřeba vybrat takovou knihovnu resp. knihovny, které budou umět pracovat s grafy a budou moci uživateli zprostředkovat vizuální náhled na graf. Dále je nezbytně nutné, aby tato knihovna resp. knihovny byly kompatibilní s prostředím RapidMiner.

V kap. 5.1 bylo popsáno šest knihoven pro práci s grafy. Některé (např. JUNG) umí zpracovávat logické grafy a zvládají i jejich vizualizaci. Jiné knihovny jsou stavěné pouze na jeden z úkolů (zpracování grafu a vizualizaci).

Knihovna yFILES nepřichází v úvahu, protože není open source a tudíž nemůže být distribuována s prostředím RapidMiner, které je poskytováno pod licencí AGPL.

Tab. 5.1: Porovnání vlastností

	Open source	Spolupráce s jinými knihovnami	Pokročilá vizualizace	Logické grafové struktury	Lib. Objekt jako vrchol grafu	Pokročilé vyhledávací algoritmy	Exportování grafů
yFILES	✗	✗	⚠	⚠	✗	✗	✗
JUNG	✓	✗	⚠	✓	✓	✓	✗
Annas	✓	✗	⚠	✓	✓	✓	⚠
G	✓	✗	⚠	✗	✗	✗	✗
JGraph	✓	✗	✓	✗	✓	✗	✗
JGraphT	✓	⚠	✗	✓	✓	✓	✓

Legenda: ✓ - obsahuje, ⚠ - obsahuje omezeně, ✗ - neobsahuje

Dle tab. 5.1 se jeví, že nejlépe umí pracovat s logickými grafovými strukturami knihovna JGraphT. Tato knihovna ovšem neumí vytvořený graf vizualizovat. Lepšími kandidáty se tudíž zdají být knihovny JUNG a Annas, které umí grafovou strukturu vizualizovat.

Ovšem tyto knihovny nedosahují úrovně knihovny JGraphT ve zpracovávání logických grafů. Naštěstí JGraphT umí omezeně spolupracovat s knihovnou pro vizualizaci JGraph prostřednictvím dříve zmíněné třídy `JGraphModelAdapter<V,E>`. Vhodným řešením se zdá být využití kombinace dvou knihoven JGraphT a JGraph.

5.3 Shrnutí

V této kapitole bylo popsáno šest knihoven pro práci s grafovými strukturami. Knihovna JGraphT je navržena pouze na práci s logickými grafy, nikoliv pro jejich vizualizaci. Oproti tomu G a JGraph slouží pouze pro zobrazování grafů. JUNG, yFILES a Annas umí pracovat s grafy a umí je také vizualizovat.

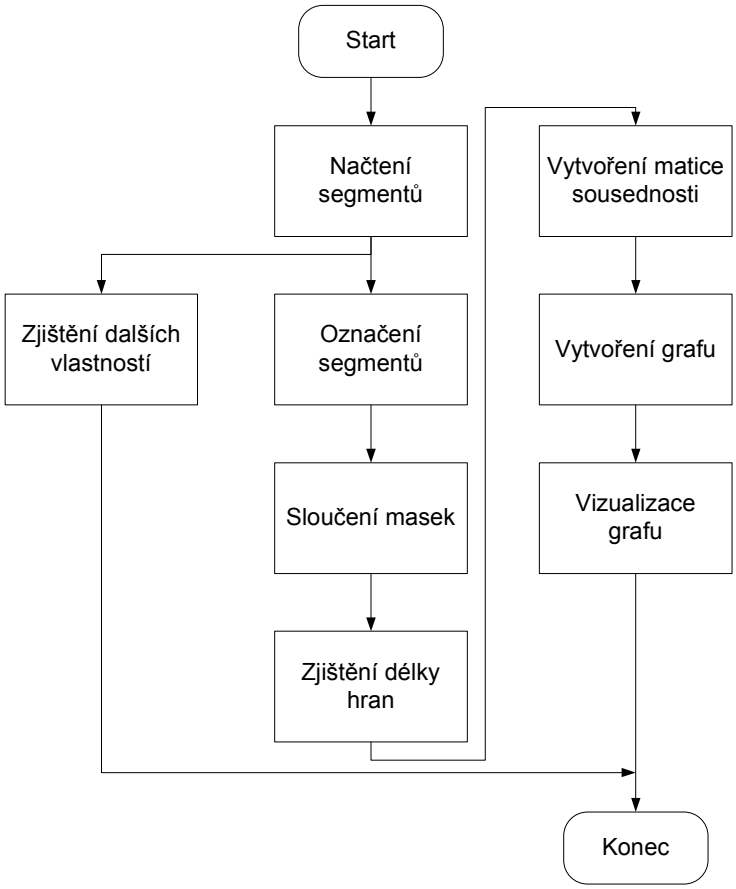
Knihovna yFiles je placená. Ostatní knihovny jsou distribuovány pod různými licencemi open source (BSD, GNU LGPL a GNU GPL).

Nejlepší z popsaných se jeví být kombinace knihoven JGraphT (práce s grafovými strukturami) a JGraph (vizualizace). Obě tyto knihovny jsou vynikající a ve svém poli působnosti patří mezi nejlepší. Spolupráce těchto knihoven je podpořena funkcí knihovny JGraphT, která umožňuje převést grafovou strukturu vytvořenou knihovnou JGraphT do vizualizované struktury grafu knihovnou JGaph.

6 Návrh algoritmu

6.1 Obecný postup

Blokové schéma obecného postupu, dle kterého se bude postupovat je zobrazen na obr. 6.1. Nejdříve se načtou masky jednotlivých segmentů z předcházejícího operátoru, který provede segmentaci vstupního obrazu a vytvoří již zmíněné masky.



Obr. 6.1 Obecný postup

Následně se masky označí příslušnou barvou. Tato barva poslouží pro identifikaci segmentu ve sloučeném obraze, který bude vytvořen v následujícím kroku. Dále budou masky segmentů sloučeny do jednoho obrazu, který bude znázorňovat rozložení segmentů v obraze. Tento obraz bude velice podobný obr. 4.1.

V kroku „Zjištění délky hran“ bude zjištěna délka společné hrany všech segmentů. V neposlední řadě se vytvoří matice sousednosti z délek hran, které byly zjištěny v kroku

„Zjištění délky hran.“ Pomocí této matice se vytvoří „logický“ graf a provede se následná vizualizace toho grafu, obojí za pomoci knihoven JGraphT a JGraph, které byly vybrány v kap. 5.

Také se vypočítají další vlastnosti jednotlivých segmentů. Na těchto výpočtech nezávisí žádný jiný krok toho postupu, takže mohou proběhnout téměř kdykoli. Vhodné tedy bude, když budou probíhat nezávisle na ostatních krocích.

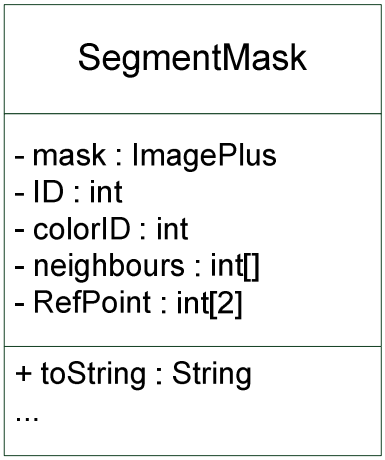
6.2 Datové typy

Nejdříve je nutné vytvořit datové typy, které ponesou informace potřebné pro vytvoření grafu a další zpracování. Tyto datové typy musejí být dva. Jeden ponese informace pouze o jednom segmentu a druhý o obraze resp. o všech segmentech v obraze. Jsou to tyto:

- SegmentMask,
- ImagePlusSegmentedIOObject.

6.2.1 SegmentMask

Diagram třídy (class diagram) popisující datový typ `SegmentMask` je zobrazen na obr. 6.2. Zde jsou vidět jednotlivé proměnné, které tento datový typ ponese. Všechny tyto proměnné budou privátní a přistupovat se k nim bude přes metody `set` a `get`. Tento datový typ popisuje jeden segment obrazu.



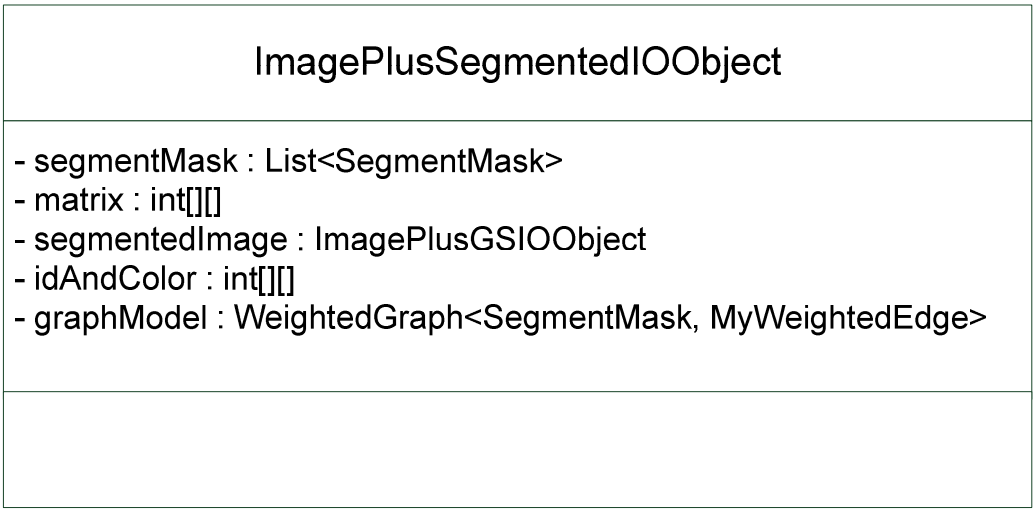
Obr. 6.2 SegmentMask - datový typ

Proměnná `mask` – maska segmentu, v sobě nese černobílý obraz o stejných rozměrech jako původní obraz. Tento obraz je celý černý až na region obsahující příslušný segment (tento region je vyplněn bílou barvou). `ID` nese identifikační číslo segmentu, toto číslo je jedinečné a zastupuje každý segment. Proměnná `colorID` v sobě nese informaci o barvě, která byla segmentu přiřazena. Pole `neighbours` obsahuje údaje o délce společné hranice s každým segmentem. Dvouprvkové pole `RefPoint` obsahuje údaj o poloze segmentu v obraze.

O metodě `toString` bude pojednáno později.

6.2.2 ImagePlusSegmentedIOObject

Na obr. 6.3 je vidět diagram třídy datového typu `ImagePlusSegmentedIOObject`. Jako u předchozího datového typu (kap. 6.2.1) jsou také zde proměnné pouze privátní a přístup k nim je možný pouze přes metody `set` a `get`. Tento datový typ nese informace o segmentovaném obrazu jako celku.



Obr. 6.3 ImagePlusSegmentedIOObject - datový typ

Seznam `segmentMask` v sobě nese všechny segmenty typu `SegmentMask` (kap. 6.2.1) obsažené v příslušném obraze, který byl podroben segmentaci. Matice `matrix` je matice sousednosti pro daný segmentovaný obraz. V proměnné `segmentedImage` je uložen tzv. sloučený obraz (vznikne sloučením všech masek daného obrazu). Matice `idAndColor` je dvousloupcová. Na každém řádku je uložena dvojice `ID` a `colorID` pro všechny segmenty. Tato matice slouží pro rychlejší vyhledávání segmentů.

Proměnná `graphModel` obsahuje „logický“ graf vytvořený pomocí knihovny JGraphT. Jako datový typ této proměnné byl zvolen `WeightedGraph<V,E>`. Pomocí tohoto datového typu lze vytvořit ohodnocený graf tzn., může mít ohodnocené vrcholy i hrany. Ve vrcholech může být jako ohodnocení uložen jakýkoliv objekt (např. `String`, `Integer`, `Double` nebo dokonce `SegmentMask`). V tomto případě v každém vrcholu grafu leží jeden segment (`SegmentMask`). Dále každá hrana bude ohodnocena číslem datového typu `int`, které bude odpovídat délce společné hranice uzlů, jež jsou propojeny touto hranou. O tom proč je jako hrana použito `MyWeightedEdge` bude pojednáno později.

6.3 Operátory

Z postupu znázorněného na blokovém schématu (obr. 6.1) by nebylo vhodné vytvořit pouze jeden operátor. Vhodnější bude seskupit některé bloky, které k sobě logicky patří, do jednoho operátoru. Takto bude vhodné vytvořit operátory tři. Jeden bude vytvářet sloučený obraz (bloky „Načtení segmentů,“ „Označení segmentů,“ „Sloučení masek“), druhý bude počítat další vlastnosti segmentů (blok „Zjištění dalších vlastností“) a třetí bude vytvářet grafovou strukturu (bloky „Zjištění délky hran,“ „Vytvoření matice sousednosti,“ „Vytvoření grafu,“ „Vizualizace grafu“).

6.3.1 Výpočet dalších vlastností

Do operátoru pro výpočet vlastností jednotlivých segmentů Statistics Roi Operator (vkládá se do operátoru Feature Extractor) byly přidány funkce pro výpočet čtyř vlastností:

- velikost,
- obvod,
- nekompaktnost,
- podlouhlost.

Tyto vlastnosti pomohou lépe identifikovat jednotlivé segmenty (např. v sekvenci obrázků pomohou určit, zda jeden objekt z jednoho snímku se nachází na dalším snímku).

Velikost

Velikost segmentu může být použita např. pro zjištění, jestli se objekt vzdaluje či nikoliv. Jsou dvě možnosti jak zapsat velikost segmentu:

- absolutně (počet pixelů obsažených v segmentu),
- poměrově (počet pixelů segmentu ku počtu pixelů celého obrazu). [21]

Velikost vyjádřená poměrově se jeví být lepší, protože lze okamžitě vidět, jakou část obrazu segment zabírá.

Výpočet proběhne následovně: Z příslušné masky se zjistí počet bílých pixelů (hodnota pixelu je 255) a tato hodnota se podělí součinem šířky a délky masky.

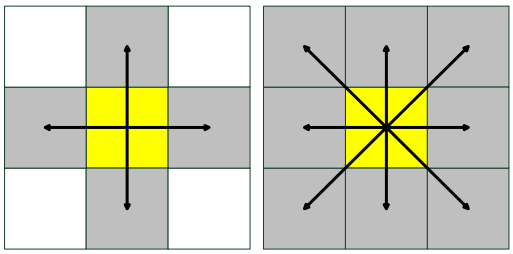
Obvod

Obvod znamená: počet pixelů ležících na hranici zkoumaného objektu [21]. Obvod může být:

- vnitřní (obvod je počet hraničních pixelů patřící objektu, jehož obvod je počítán),
- vnější (obvod je počet hraničních pixelů nepatřící objektu, jehož obvod je počítán, avšak tyto pixely musí sousedit s tímto objektem).

Dále se také může obvod dělit podle použitého okolí:

- 4okolí – ukázka okolí je na obr. 6.4, pixel (žlutý) sousedí pouze s pixely označenými šedě (tedy při zjišťování sousedů může jít pouze směry, které jsou vyznačeny na obrázku),
- 8okolí – pro zjišťování sousedů může centrální pixel používat oproti 4okolí navíc další čtyři směry vyznačené na obr. 6.4.



Obr. 6.4 4okolí (vlevo), 8okolí (vpravo)

V operátoru bude počítán vnitřní obvod s použitím 4okolí. Postup bude následující: projde se každý pixel masky segmentu, a pokud je tento pixel bílý, tak se inkrementuje čítač za předpokladu, že je splněna jedna z podmínek – zkoumaný pixel leží na okraji obrazu, nebo zkoumaný pixel má ve svém 4okolí alespoň jeden černý pixel (hodnota 0).

Nekompaktnost

Nekompaktnost značí míru podobnosti objektu k ideálnímu kruhu. Vypočítá se podle následujícího vzorce:

$$N = \frac{o^2}{S}, \tag{6.1}$$

kde N je nekompaktnost, o je obvod objektu a S je velikost objektu (absolutní – v pixelech). [21]

V operátoru se nekompaktnost počítá dle vzorce (6.1) a s použitím dříve zjištěných hodnot (obvod a velikost).

Podlouhlost

Podlouhlost vyjadřuje poměr mezi délkou a šířkou pravoúhelníku opsanému objektu. Vypočítá se dle následujícího vztahu:

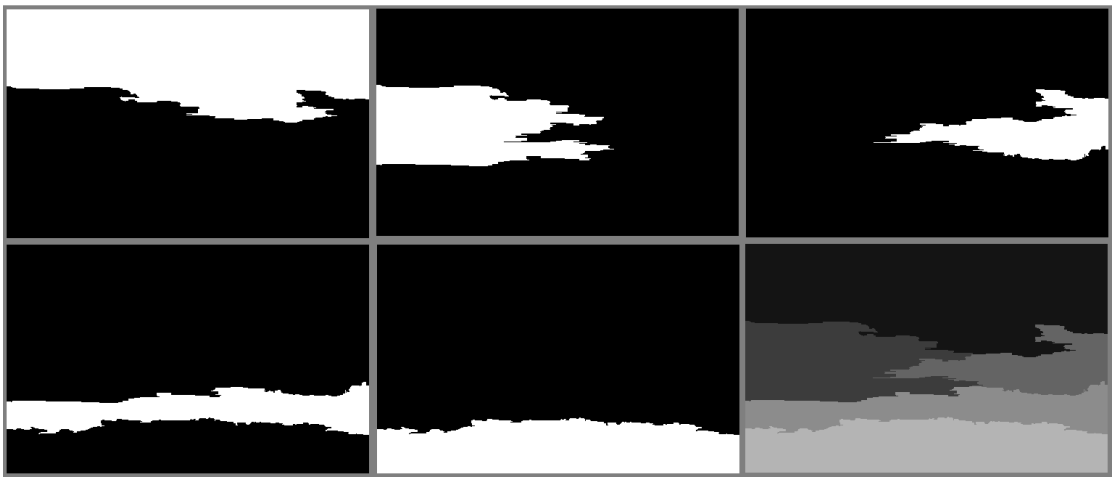
$$P = \frac{d}{š}, \tag{6.2}$$

kde P je podlouhlost, d je délka a $š$ šířka opsaného pravoúhelníka. [21]

V operátoru se zjistí šířka a délka objektu a dosadí se do vzorce (6.2).

6.3.2 Create Grayscale Mask

Dalším operátorem, který bude popsán je Create Grayscale Mask. Tento operátor vytváří ze vstupních masek sloučený obraz. Vstupní masky i sloučený obraz jsou vidět na obr. 6.5 (pět vstupních masek; sloučený obraz je vpravo dole).



Obr. 6.5 Masky a sloučený obraz

Operátor funguje následujícím způsobem:

1. Načtení instance třídy `ImagePlusSegmentedIOObject`, která v sobě nese všechny segmenty (tyto segmenty již byly opatřeny `ID` při segmentaci).

2. Inicializace proměnných, které budou používány.

3. Přiřazení barvy každé masce z rozsahu 20-200, zapsání dvojic `ID` a barva (`colorID`) do matice `idAndColor`.

4. V sloučeném obraze se obarví region, který vytyčuje každá maska, příslušnou barvou.

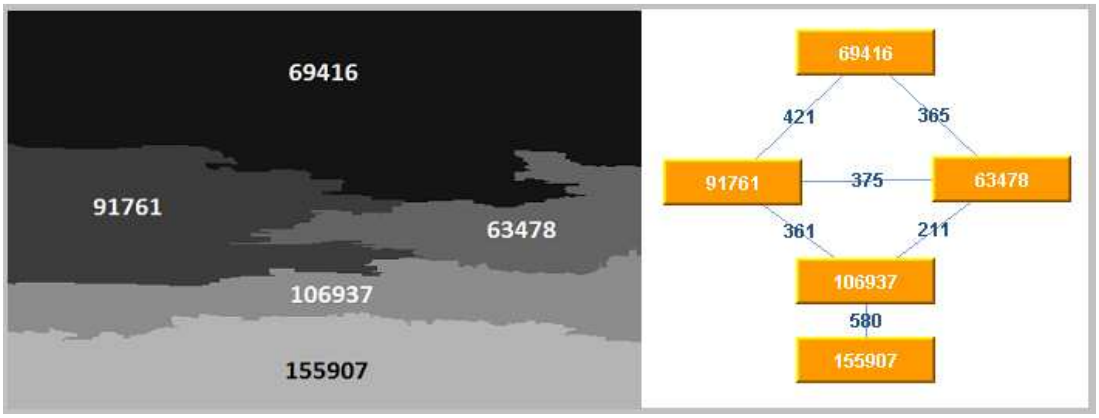
5. Do instance třídy `ImagePlusSegmentedIOObject` se vloží sloučený obraz a matice `idAndColor` a tato instance třídy se předá na výstup.

Poznámka k bodu 3: Přiřazení barev segmentům je kritické. Přiřazená barva musí být jedinečná, tzn. žádný jiný segment nesmí mít přiřazenou stejnou barvu (tato jedinečnost barev usnadní práci při zjišťování délky společné hrany). Barvy jsou voleny z rozsahu 20-200, aby nedošlo k záměně či dezinterpretaci (barvy černá a bílá jsou používány

v maskách). Přiřazení barev je naprogramováno, tak aby rozložení těchto barev rovnoměrně pokrývalo zvolený rozsah.

6.3.3 Graph Maker

V neposlední řadě bude popsán operátor Graph Maker. Tento operátor vytváří grafovou strukturu segmentů jednoho snímku. Příklad sloučeného obrazu a grafu, který tento operátor vytváří je na obr. 6.6 (ohodnocení vrcholů odpovídá `ID` jednotlivých masek a ohodnocení hran odpovídá společné délce hranice v pixelech).



Obr. 6.6 Sloučený obraz s popisem (vlevo), graf smínku (vpravo)[2]

Operátor `Graph Maker` funguje následujícím způsobem:

1. Načtení instance třídy `ImagePlusSegmentedIOObject` ze vstupu operátoru.
2. Inicializace proměnných, které budou používány.
3. Vytvoření matice sousednosti.

   K tomuto úkolu lze přistupovat dvěma různými způsoby:

   - sledování hrany každého segmentu v sloučeném obraze a zjišťování sousedů v každém pixelu této hranice,
   - procházení sloučeného obrazu po řádcích a zjišťování sousedů zkoumaného pixelu.

[2] Čísla na sloučeném obrázku (vlevo) byla přidána pro přehlednost, nejsou tedy součástí výstupu.

První způsob bude dávat vždy dobré výsledky, ale bude hůře implementovatelný. Oproti tomu druhý způsob nemusí dávat vždy dobré výsledky, pokud nebude správně ošetřen, ale je jednodušší na implementaci.

Byl tedy zvolen druhý způsob, který bude fungovat následujícím způsobem: Sloučený obraz bude procházen po řádcích a u zkoumaného pixelu (o souřadnicích [*x*,*y*], kde *x* resp. *y* je sloupcová resp. řádková souřadnice) se budou zjišťovat pouze dva sousedé – pixel vpravo (souřadnice [*x*+1,*y*]) a pixel dole (souřadnice [x,y+1]). Zbylí dva sousedé z 4okolí nemusí být zjišťovány, protože tyto pixely již byly projity a přiřazeny.

Pro zjišťování sousedů, jak již bylo řečeno, bude použito 4okolí. Dále kvůli zajištění souměrnosti matice sousednosti (viz kap. 2.3.2) a kvůli stejnorodosti výsledků bude použita nejmenší hranice mezi sousedy (každý pixel může být jednomu segmentu započítán pouze jednou jako sousední). Tedy ve většině případů již nebude platit rovnost:

$$O_k = \sum_{i=0}^{C-1} s_{ki}\,, \tag{6.3}$$

kde O_k je obvod k-tého segmentu, C značí počet sousedů segmentu a s_k je délka hranice segmentu s jeho sousedem.

4. Vytvoření grafu.

   Tento graf bude vytvořen pomocí knihovny JGraphT. V každém vrcholu grafu je uložen segment příslušného obrazu (datový typ `SegmentMask`). Knihovna JGraphT k zobrazování hodnot vrcholů používá metodu `toString` objektu uloženého ve vrcholu. Proto byla stejnojmenná metoda vytvořena ve třídě `SegmentMask` (přesněji byla překryta metoda `toString` třídy `Object`). Tato metoda vrací `ID` jednotlivého segmentu.

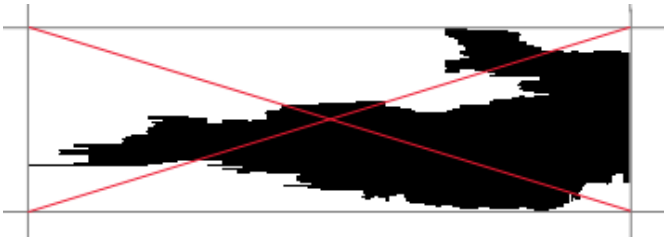
   Dále budou vytvořeny hrany dle již vytvořené matice sousednosti. Při vytváření instance grafu (`WeightedGraph`) byla použita mj. třída `MyWeightedEdge`, která byla odvozena od třídy `WeightedEdge`, která jako ohodnocení vypisovala pouze,

jaké vrcholy jsou spojeny. V třídě `MyWeightedEdge` byla překryta metoda `toString` tak, aby vracela ohodnocení příslušné hrany.

5. Zjištění referenčních bodů.

   Před vlastní vizualizací se zjistí body, kde budou jednotlivé vrcholy zobrazeny. Tyto body se vypočítají jako střed pravoúhelníka opsaného segmentu (strany tohoto pravoúhelníka jsou rovnoběžné se stranami obrazu). Získání tohoto referenčního bodu je znázorněno na obr. 6.7 (referenční bod leží na průsečíku červených linií).

   Tyto body se pak uloží do proměnné `RefPoint`, která je součástí datového typu `SegmentMask`. Každý bod se tedy uloží do vlastností příslušného segmentu.



Obr. 6.7 Zjištění referenčního bodu

6. Vizualizace grafu.

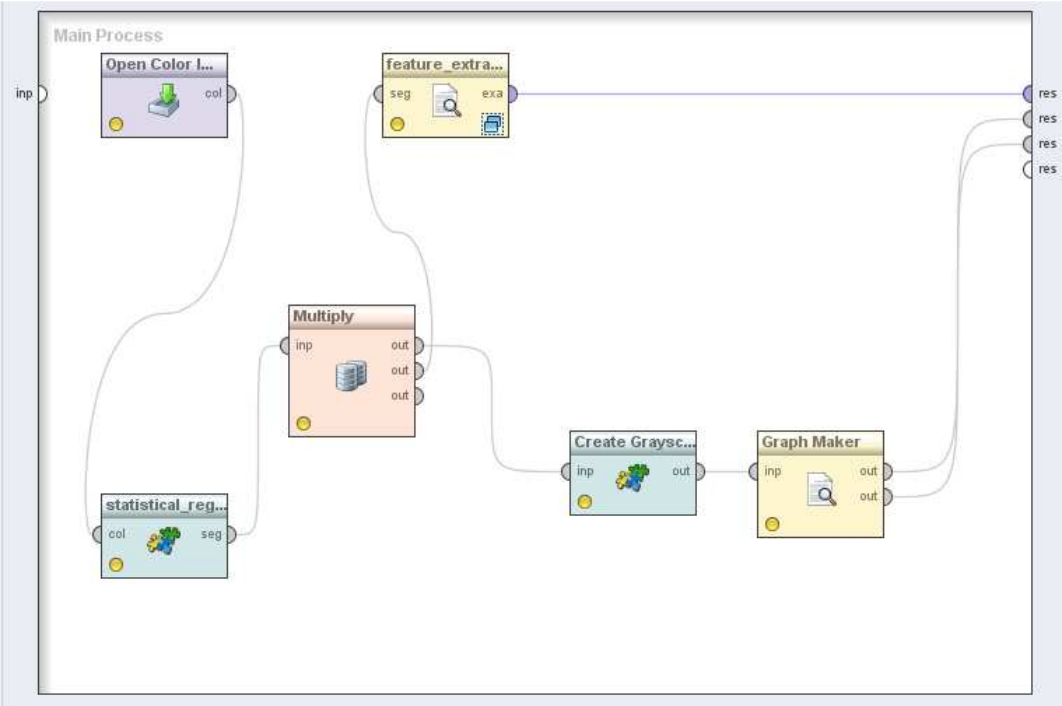
   Vizualizace probíhá pomocí knihovny JGraph a také pomocí jedné třídy z knihovny JGraphT `JGraphModelAdapter`. Také zde byla implementována funkce pro změnu měřítka zobrazení grafu. Uživatel zadá pouze, na jakou šířku grafu chce změnit (normalizovat) měřítko a dle zadané šířky se vypočítá i nová výška grafu. A podle těchto dvou hodnot bude normalizována vizualizace celého grafu.

7. Odeslání dat na výstup.

   Na výstup bude poslána instance třídy `ImagePlusSegmentedIOObject`, která bude obsahovat stejné hodnoty jako na vstupu operátoru a navíc budou přidány o informace zjištěné pomocí tohoto operátoru. Dále bude předána na výstup instance třídy `GraphIOObject`, která v sobě ponese pouze vizualizovaný graf.

6.4 Zapojení

Nejdříve se načte barevný obraz pomocí operátoru Open Color Image. Výstup tohoto operátoru je přiveden na vstup dalšího (Statistical region mergin), který se stará o segmentaci. Následující operátor je Multiply, který zajišťuje klonování dat z předchozího operátoru. Výstup tohoto operátoru je připojen na Feature Extractor (spolu v operátory zařazeny v něm zajišťuje výpočet vlastností segmentů) a na operátor Create Grayscale Mask (viz kap. 6.3.2). Posledním operátorem tohoto řetězce je Graph Maker (viz kap. 6.3.3). Ukázka tohoto zapojení je na obr. 6.8.



Obr. 6.8 Zapojení[3]

6.5 Výstup

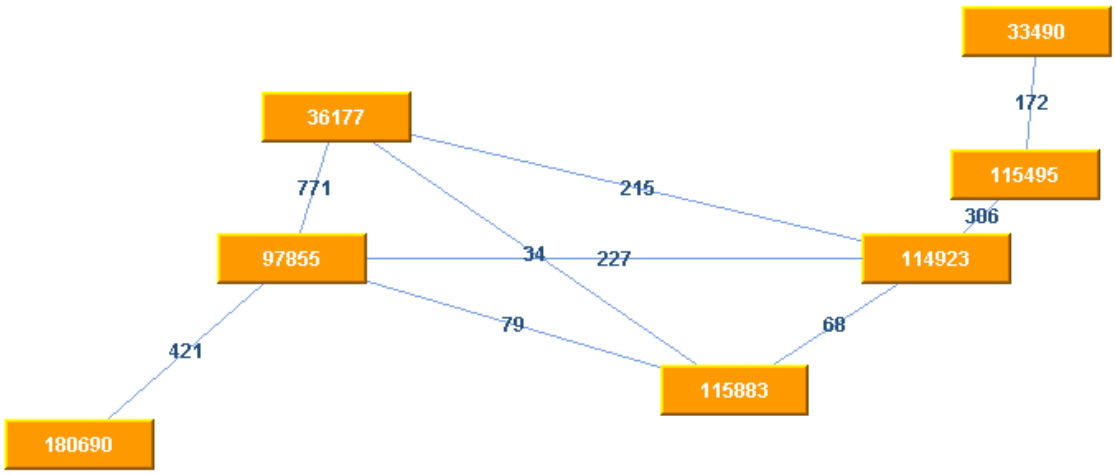
Vykreslení výstupů mají v prostředí RapidMiner na starosti tzv. Renderery. Proto byly vytvořeny Renderery pro `ImagePlusSegmentedIOObject` a `GraphIOObject`. Operátor Feature Extractor má na výstupu tzv. `example set`, tato struktura se uživateli zobrazuje jako přehledná tabulka.

[3] Operátor Multiply je součástí prostředí RapidMiner. Vytváření operátorů Open Color Image, Statistical region merging a Feature Extractor není součástí této práce. Avšak pro zajištění chodu navrženého postupu byly použity.

Ze vstupních masek byl vytvořen tzv. sloučený obraz. Vstupní i sloučený obraz je vidět na obr. 6.9.

Obr. 6.9 Originální a sloučený obraz

Dále byl také vytvořen graf segmentů. Tento graf je plně interaktivní (uživatel může libovolně měnit vizualizaci grafu) díky knihovně JGraph. Rozložení vrcholů v grafu odpovídá rozložení segmentů v obraze. Tento graf je vidět na následujícím obrázku (obr. 6.10).



Obr. 6.10 Graf

6.6 Testování

Byly provedeny tři testy pro zjištění správnosti navrženého postupu. Vstupní obraz pro každý test má rozměry 100x100 px (pixel).

První test

Jako vstupní obraz byl použit obraz se dvěma obdélníky o rozměrech 20x40 px (příloha A.1). Tyto obdélníky nemají žádný společný bod, tuto skutečnost znázorňuje vytvořený graf (příloha A.3). Graf také udává, že délka společné hranice (obdélníka a pozadí) je 116 (shoduje s vypočteným obvodem), toto číslo je správné, protože žádný pixel nesmí být započítán vícekrát jako sousední k jednomu segmentu. Tudíž obvod bude o 4 menší než prosté sečtení délek jeho stran (každý roh obdélníka patří do dvou stran). Příloha A.2 obsahuje sloučený obraz.

Dále podlouhlost každého obdélníku je 0,5, tato hodnota je správná (poměr 20/40). Relativní velikost je 0,08 ($(40 \cdot 20)/(100 \cdot 100)$). Vypočtená nekompaktnost je 16,82, tato hodnota je správná a byla vypočtena dle vzorce (6.1):

$$N = \frac{o^2}{S} = \frac{116^2}{20 \cdot 40} = \frac{13456}{800} = 16{,}82.$$

Druhý test

Vstupním obrazem (příloha A.4) tohoto testu je obraz podobný obrazu z testu prvního, rozdíl je v tom, že zmíněné obdélníky mají společnou stranu (její délka je 40 px). Graf (příloha A.6) podle předpokladu udává, že délka společné hranice dvou obdélníků je 40. Dále pozadí a každý obdélník mají délku společné hranice 78 (pouze dva rohy každého obdélníka mohli být započteny dvakrát, ale nebyly). V tomto testu byly použity stejné objekty ve vstupním obraze jako v prvním testu, proto mají jejich vlastnosti stejné hodnoty a není tedy třeba je uvádět.

Příloha A.5 obsahuje sloučený obraz k tomuto testu.

Třetí test

Tento test je komplexnější než předchozí dva testy. Je použit složitější vstupní obraz (příloha A.7). Je zde obsaženo více objektů a některé tyto objekty se překrývají. Ovšem za použití okótovaného vstupního obrazu (příloha A.8) a sloučeného obrazu (příloha A.9, pro lepší orientaci je lepší použít popsaný sloučený obrázek – příloha A.10) lze zjistit, že výsledný graf (příloha A.11) je v pořádku. Např. pro délku hranice mezi segmenty s `ID` 41 a 6832: $20 + 2 \cdot 10 - 2 = 38$. Tato hodnota odpovídá hodnotě zapsané

v grafu. Dalším příkladem je výpočet délky hranice mezi segmenty s `ID` 1861 a 41: 20 + 40 − 1 = 59. I tato hodnota odpovídá hodnotě zapsané v grafu.

Vypočtená nekompaktnost N pro segment s `ID` 41 je 48,05. Tato hodnota je správná a počítala se dle vzorce (6.1):

$$N = \frac{o^2}{S} = \frac{310^2}{2000} = \frac{96100}{2000} = 48{,}05,$$

kde o je obvod příslušného segmentu a S velikost segmentu v pixelech.

Vypočtená podlouhlost P pro segment s `ID` 41 je 2,5. Tato hodnota je správná a byla vypočtena podle vzorce (6.2):

$$P = \frac{d}{\check{s}} = \frac{100}{(20 + 20)} = 2{,}5,$$

kde d je délka a $\check{s}$ šířka opsaného pravoúhelníka příslušného segmentu.

6.7 Shrnutí

Pro popis obrazu podrobeného segmentaci byly navrženy dvě datové struktury. První je `SegmentMask`, takto struktura popisuje jeden segment ze segmentovaného obrazu. Další je `ImagePlusSegmentedIOObject`, ta popisuje segmentovaný obraz jako celek.

Dále byly představeny vlastnosti (velikost, obvod, nekompaktnost a podlouhlost), které se budou počítat pro každý segment.

Byly také navrženy dva operátory: jeden vytváří ze vstupních masek sloučený obraz (Create Grayscale Mask) a druhý vytváří grafovou reprezentaci segmentovaného obrazu (Graph Maker) – tento operátor vytváří jak logickou grafovou strukturu, tak i vizualizovanou grafovou strukturu.

V posledním avšak neméně důležitém kroku bylo provedeno testování navržených algoritmů. Testy byly prováděny na třech obrázcích (dva jednodušší a jeden složitější). Tyto testy skončily dle předpokladů úspěšně a potvrdily funkčnost navrženého postupu.

7 Závěr

Nejdříve byl vysvětlen pojem segmentace a posléze byly popsány různé segmentační metody. Dále byla také stručně zmíněna teorie grafů. Zde byly vysvětleny základní pojmy z teorie grafů, představeny některé metody pro prohledávání grafu a ukázány možné způsoby reprezentace grafu.

Byl proveden průzkum knihoven zabývajících se prací s grafy a jejich vizualizací. V rámci tohoto průzkumu bylo představeno šest knihoven, které jsou vhodné pro pokročilé zpracování grafových struktur. Z těchto knihoven byly vybrány dvě, které svými vlastnostmi převyšovaly ostatní. Jsou to tyto: JGraphT, tato knihovna zpracovává logické grafové struktury, a JGraph, tato knihovna vizualizuje grafové struktury.

V dalším kroku byly navrženy a implementovány dvě datové struktury pro popis segmentovaného obrazu. První z nich popisuje jednotlivé segmenty a druhá popisuje segmentovaný obraz jako celek.

Také byly navrženy a zrealizovány dva operátory pro prostředí RapidMiner, které mají za úkol naplnit tyto datové struktury vypočtenými údaji. První operátor má za úkol vytvořit sloučený obraz ze vstupních masek. Druhý operátor vytváří ze sloučeného obrazu graf a také jeho vizualizaci. Dále byly navrženy a naprogramovány postupy pro vypočtení některých vlastností segmentů.

V neposlední řadě byly provedeny tři testy. Hodnoty z každého testu se porovnávali s teoretickými hodnotami a nebyla zjištěna žádná odchylka. Díky těmto testům byla dokázána správnost návrhu i následné realizace postupů.

Literatura

[1] ANDERSON, R. The Universal Photographic Digital Imaging Guidelines. *Updig.* [Online] 2006. http://www.updig.org/.

[2] GREENBERG, I. *Processing: creative coding and computational art.* New York : Apress, 2007. ISBN: 1-59059-617-X.

[3] ŠONKA, M.; HLAVÁČ, V.; BOYLE, R. *Image Processing, Analysis, and Machine Vision, 3rd Edition.* Toronto : Cengage-Engineering, 2007. ISBN: 049508252X.

[4] HLINĚNÝ, P. Teorie grafů. *Fakulta Informatiky Masarykovy univerzity.* [Online] 14. Srpen 2008. [Citace: 6. Listopad 2010.] www.fi.muni.cz/~hlineny/Vyuka/GT/Grafy-text07.pdf.

[5] ŠEDA, M. Teorie grafů. *ÚAI FSI VUT v Brně.* [Online] 2003. [Citace: 6. Listopad 2010.] www.uai.fme.vutbr.cz/~mseda/TG03_MS.pdf.

[6] BOLLOBÁS, B. *Modern Graph Theory.* New York : Springer Science + Bussines Media Inc., 1998. ISBN: 0-387-98491-7.

[7] DIESTEL, R. *Graph Theory, Third Edition.* New York : Springer-Verlag Inc., 2006. ISBN: 3-540-26183-4.

[8] BROUSEK, J. a další. Orientované grafy, matice a počet koster. [autor knihy] Jan Brousek, a další. *Diskrétní matematika.* Plzeň : Katedra matematika FAV, Západočeská univerzita v Plzni, 2003.

[9] RAPID-I. *RapidMiner Tutorial.* [internet] Dortmund, Německo : Rapid-I, 1. Říjen 2009. http://rapid-i.com/.

[10] RAPID-I . GNU Affero General Public License (AGPL). *Rapid-I Report The Future.* [Online] 19. Listopad 2007. [Citace: 17. Listopad 2010.] http://rapid-i.com/content/view/29/215/lang,en/.

[11] YWORKS. yFILES. *yWorks.* [Online] yWorks, 2011. [Citace: 3. Duben 2011.] http://www.yworks.com/en/index.html.

[12] THE JUNG FRAMEWORK DEVELOPMENT TEAM. JUNG. *JUNG.* [Online] University of California, 24. Leden 2010. [Citace: 3. Duben 2011.] http://jung.sourceforge.net/index.html.

[13] JUNG PROJECT. Project License. *JUNG2.* [Online] University of California, 24. Leden 2010. [Citace: 3. Duben 2011.] http://jung.sourceforge.net/site/license.html.

[14] ANNAS PROJECT. Welcome to annas. *Annas.* [Online] Annas Project, 2011. [Citace: 6. Duben 2011.] https://sites.google.com/site/annasproject/Home.

[15] FREE SOFTWARE FOUNDATION, INC. GNU General Public License. *GNU Operating System.* [Online] Free Software Foundation, Inc., 2007. [Citace: 6. Duben 2011.] http://www.gnu.org/licenses/gpl.html.

[16] GEOSOFT. G 2D Graphics Library and Rendering Engine for Java. *GeoSoft.* [Online] GeoSoft, December 2009. [Citace: 8. Duben 2011.] http://geosoft.no/graphics/.

[17] FREE SOFTWARE FOUNDATION, INC. GNU Lesser General Public License. *GNU Operating System.* [Online] Free Software Foundation, Inc., 29. Červen 2007. [Citace: 8. Duben 2011.] http://www.gnu.org/copyleft/lesser.html.

[18] BENSON, D.; ALDER, G. JGraphX (JGraph 6) User Manual. *JGraph. Visualize Everything.* [Online] JGraph, 3. Listopad 2010. [Citace: 21. Listopad 2010.] http://www.jgraph.com/doc/mxgraph/index_javavis.html.

[19] NAVEH, B. Welcome to JGraphT - a free Java library. *JGraphT.* [Online] JGraphT, 2005. [Citace: 10. Duben 2011.] http://www.jgrapht.org/.

[20] JGRAPHT. Javadoc. *JgraphT.* [Online] JgraphT. http://www.jgrapht.org/javadoc/.

[21] HORÁK, K. Detekce a klasifikace objektů. *Aplikace počítačového vidění.* Brno : VUT v Brně, FEKT, 2010. http://www.google.cz/url?sa=t&source=web&cd=1&ved=0CBgQFjAA&url=http%3A%2F%2Fwww.uamt.feec.vutbr.cz%2Fvision%2FTEACHING%2FMAPV%2F05%2520-%2520Detekce%2520a%2520klasifikace%2520objektu.pdf&rct=j&q=detekce%20a%20klasifikace%20objekt%C5%AF&ei=q5-yTf_7E8OXOse.

Seznam použitých zkratek, veličin a symbolů

$A(G)$	Matice sousednosti (Adjacency matrix).
AGPL	Affero General Public License.
BFS	Breadth-first search - Prohledávání do šířky.
BSD	Berkeley Software Distribution.
C	Počet sousedů.
d	Délka opsaného pravoúhelníka.
DFS	Depth-first search - Prohledávání do hloubky.
G	Graf.
GML	Graph Modelling Language.
GPL	General Public License.
H	Hrana.
ID	Identifikátor.
JDK	Java Development Kit.
JUNG	Java Universal Network/Graph Framework.
$l(U)$	Aktuální odhad nejkratší vzdálenosti mezi startovacím uzlem a uzlem U.
LGPL	Lesser General Public License.
$M(G)$	Incidenční matice (Incidence matrix).
MS	MicroSoft.
N	Nekompaktnost.
o	Obvod.
O	Asymptotická složitost, notace Omikron.
P	Podlouhlost.
px	Pixel.
RGB	Red Green Blue.
s	Délka hranice.
S	Velikost.
$š$	Šířka opsaného pravoúhelníka.
URL	Uniform Resource Locator.
V	Vrchol, uzel.
W	Váha hrany.

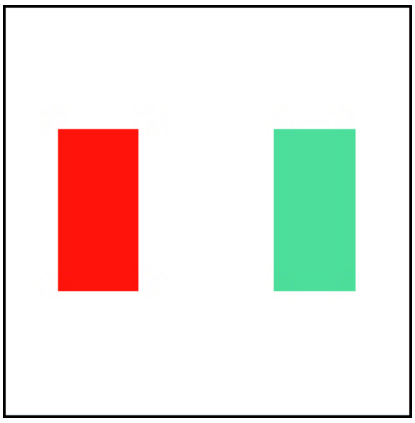
x	Sloupcová souřadnice.
XML	eXtensible Markup Language.
y	Řádková souřadnice.
$\delta(G)$	Nejnižší stupeň vrcholu.
$\Delta(G)$	Nejvyšší stupeň vrcholu.

Seznam Příloh

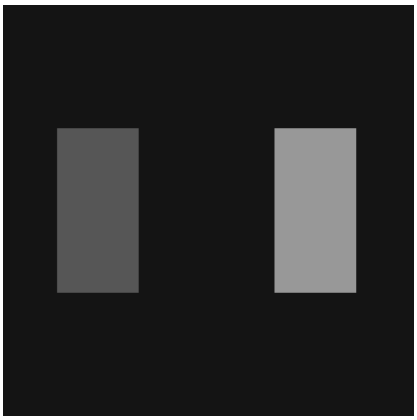
Přílohy

A Testování

Příloha A.1 První test - vstupní obraz

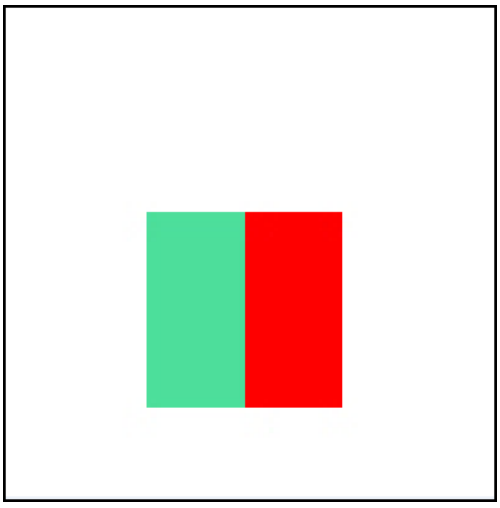


Příloha A.2 První test - sloučený obraz

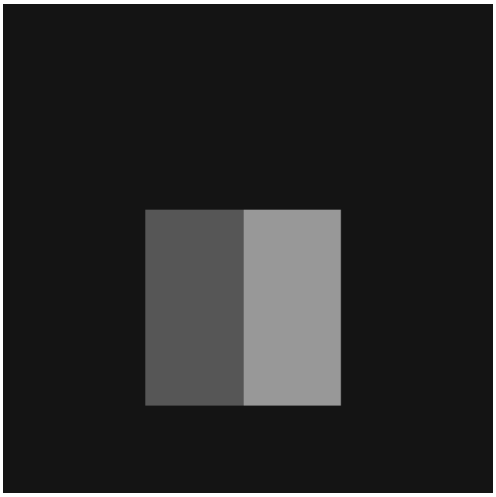


Příloha A.3 První test - graf

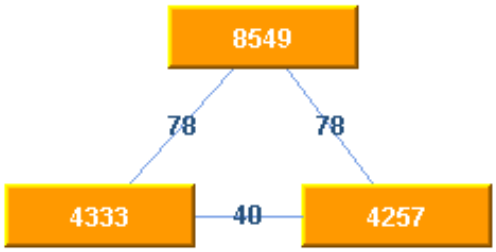
Příloha A.4 Druhý test - vstupní obraz



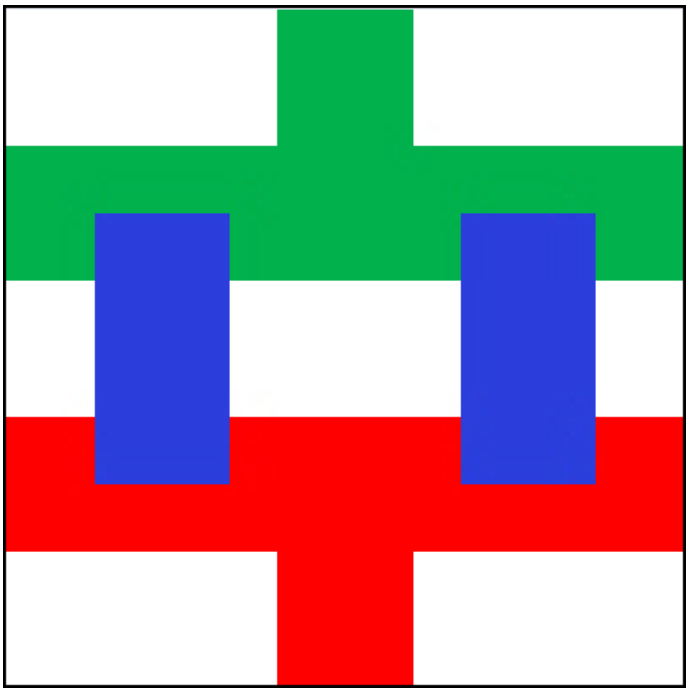
Příloha A.5 Druhý test - sloučený obraz



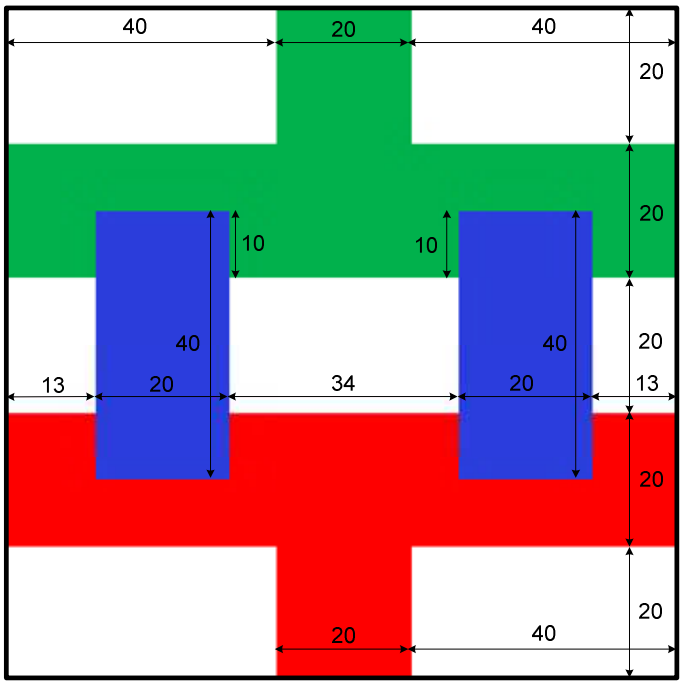
Příloha A.6 Druhý test - graf



Příloha A.7 Třetí test - vstupní obraz

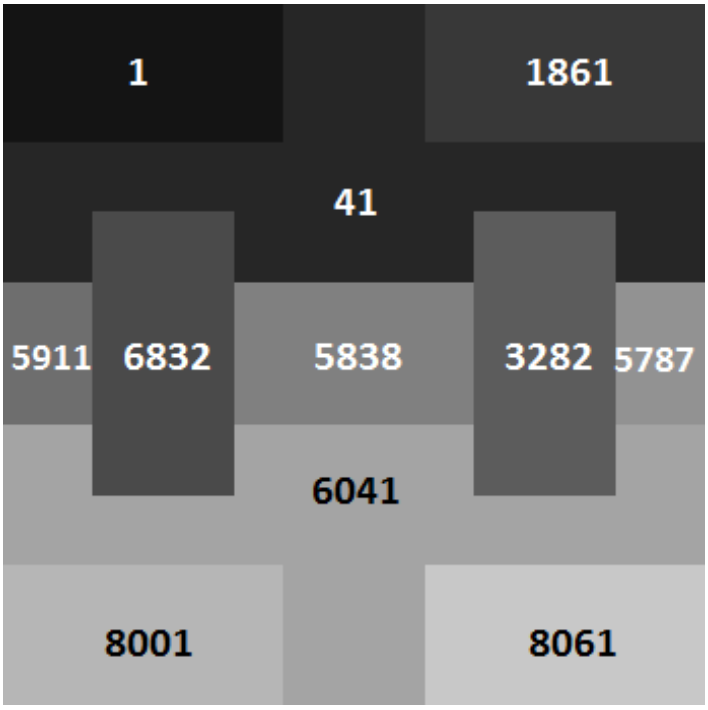


Příloha A.8 Třetí test - vstupní obraz opatřený kótami

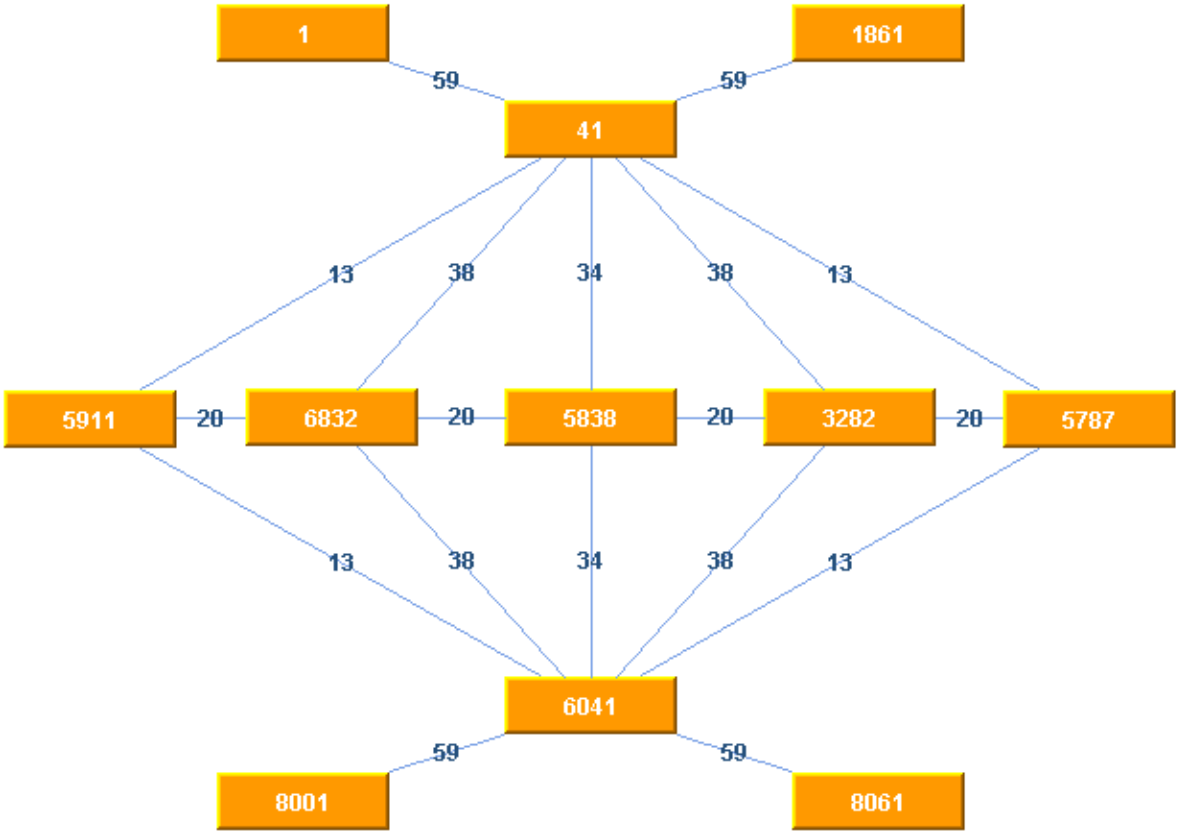


Příloha A.9 Třetí test - sloučený obraz

Příloha A.10 Třetí test - popsaný sloučený obraz



Příloha A.11 Třetí test - graf



B Obsah média

Na přiloženém médiu je celý projekt ImageProcessingExtension. Osobně jsem pracoval na třídách: CreateGrayscaleMask, GraphIOObject, GraphMaker, GraphRenderer, ImagePlusSegmentedIOObject, ImagePlusSegmentsRenderer, MyWeightedEdge, SegmentMask, StatisticsRoiOperator. Na ostatních třídách jsem se nepodílel, ale jsou přiloženy k práci, protože jsou nutné pro běh programu.

Postup pro spuštění prostředí RapidMiner se nachází na adrese:
http://spl.utko.feec.vutbr.cz/en/how-o-develop-with-rapidminer.

Příloha B.1 Obsah přiloženého média